# Adobe XAP Metadata Framework

**Version 1.0**

***February 22, 2001***

**ADOBE SYSTEMS INCORPORATED**
**Corporate Headquarters**
345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000

| Date/Rev # | Comments |
|---|---|
| | |
| | |
| | |
| | |
| | |

# Contents

# List of Tables

# 1 Preface

## 1.1 About This Document

This document describes the Adobe® eXtensible Authoring and Publishing (XAP) Metadata Framework.

This section contains information about this document, including describing how it is organized, conventions used in the document, and where to go for additional information.

## 1.2 Audience

The primary audience for this document is developers of applications that will generate, process, or manage files containing XAP metadata.

## 1.3 Assumptions

This document assumes that you are either familiar with XML and RDF, or that you will reference those specifications and related standards while reading this document.

## 1.4 How This Document Is Organized

In addition to this preface, this document consists of six chapters and an appendix, which are described below:

**Chapter 2: XAP Metadata Framework**

Explains XAP metadata and the model for how it is used, and provides a conceptual model of how metadata is created and managed. It explains the background and scope of the XAP framework, and defines basic terms and concepts. It also describes how new schema may be defined to meet needs beyond what is supported by the existing model. A number of code samples are shown to illustrate how XAP metadata is represented.

**Chapter 3: XAP RDF Data Interchange Format**

Describes how XAP uses the RDF format for data representation and how to embed metadata using XAP Packets to make it easy for applications to locate metadata in application files.

**Chapter 4: XAP Metadata Schemas**

Specifies all currently supported schema for Adobe core metadata. It also specifies the value types used for all properties, and contains the XAP Vocabularies, which specify the set of allowed values for each property. The schema tables also specify the category of each property, and lists aliases to properties in other schema.

**Chapter 5: Metadata Property Commentary**

Discusses the XAP-defined properties and describes their nature and expected use. Also, suggestions are provided for how certain properties should be handled when documents are embedded in other documents.

**Chapter 6: XAP Extensibility**

Explains the extensibility features of XAP, including how to extend schemas, add new schemas, and add private data.

**Chapter 7: Application Integration Guidelines**

Describes what applications must do to implement support for metadata, and what actions they have to perform for common application operations. It explains what media management systems need to do to support XAP metadata, and how to support the embedding of one document in another.

**Appendix: PDF and Dublin Core Schema**

Specifies the PDF and Dublin Core schemas, and specifies which properties are aliased to properties in the core XAP schemas.

## 1.5 Conventions used in this Document

XAP property names are shown in a sans-serif font; for example: xap:CreationDate.

Punctuation such as commas and periods generally conforms to the U.S. standard of being placed inside quotation marks when, for example, the quoted text ends with a URL. For example:

*The namespace is "http://ns.adobe.com/xap/1.0/g/."*

It should be obvious that the period ends the sentence, and is not a literal part of the text string or URL. In some cases where it is not so obvious, the punctuation is placed outside the quotation marks so that the quotation marks precisely designate the exact text string.

## 1.6 Where to Go for More Information

The following is a list of Internet standards on which XAP Metadata is based:

Extensible Markup Language (XML):
http://www.w3.org/XML/

Resource Description Framework (RDF):
    http://www.w3.org/RDF/

Resource Description Framework (RDF) Model and Syntax Specification:
    http://www.w3.org/TR/REC-rdf-syntax/

Dublin Core Metadata Initiative:
    http://purl.org/DC/

Web Distributed Authoring and Versioning (WebDAV):
    http://www.webdav.org/

Naming and Addressing: URIs, URLs, etc.:
    http://www.w3.org/Addressing/

XML Namespaces:
    http://www.w3.org/TR/REC-xml-names/

XML Path Language (XPath):
    http://www.w3.org/Tr/xpath

Internet Engineering Task Force (IEFT):
    http://www.ietf.org/

IETF Standard for Language element values (RFC 1766):
    http://www.ietf.org/rfc/rfc1766.txt?number=1766

ISO 639 Standard for Language Codes:
    http://www.loc.gov/standards/iso639-2/

ISO 3166 Standard for Country Codes:
    http://www.din.de/gremien/nas/nabd/iso3166ma/

# 2 XAP Metadata Framework

## 2.1 Introduction

Metadata is becoming increasingly important in the production, management, and publication of multimedia documents. Documents containing metadata can greatly increase the utility of managed assets in collaborative production workflows.

An example of metadata used in a workflow environment would be an image file that contains metadata such including the image's working title, image description, thumbnail image, and intellectual property rights data. This metadata about the file contents enables workflow users, as well as asset management systems, to use resources much more effectively. Without the appropriate metadata, it would be difficult to associate images with their file names, to locate image captions, or to determine copyright clearance to use the image. While these operations are sometimes handled by various applications using their own metadata format, it is the interchange of the metadata that is not so easy.

The importance of the XAP Metadata Framework is that it standardizes the processing and interchange of metadata, which helps to enable the full potential of collaborative workflows.

The Adobe XAP Metadata Framework defines an extensible representation that allows applications and tools to access and understand metadata about documents that they manipulate. XAP metadata defines a core set of metadata properties that are relevant for a wide range of applications including all of Adobe's authoring and publishing products, as well as for applications from a wide variety of vendors.

XAP also provides an file embedding mechanism, called a XAP Packet, that allows applications to easily locate metadata in files by simple scanning, rather than needing to parse a specific application's file format. This feature makes the metadata more easily accessible, and aids document interchange and asset management.

The XAP Metadata Framework includes the following key features:

- It provides a lightweight distributed implementation, thus avoiding a single monolithic implementation that all applications must obey.
- It accommodates a wide variety of workflows and tool environments.
- It supports extension by addition of standard schemas, by addition of private or application-specific schemas, and by updating schemas to new versions.
- It supports aliasing between multiple schemas, so it is efficient in the storage of, and access to, items that are logically equivalent (for example, PDF's pdf:Author and Dublin Core's dc:creator properties).
- It is localizable and supports Unicode and other standard international text encodings.

XAP metadata can be encoded as an XML formatted string using the W3C standard Resource Description Framework (RDF), which is described in Chapter 3, "XAP RDF Data

Interchange Format.” Ideally, the metadata string is embedded in an application file and hence appears as part of the resource data stream itself. This has the advantage that the metadata stays with the application file, even if the file is moved.

If embedding is not possible (typically because of file formats that do not accommodate extensibility), the metadata stream can be stored in a separate file that is associated by convention with the application data file. This works universally, but has the disadvantage that the metadata can be lost in a processing step if the metadata file is not kept together with the application data file.

In addition, a media management system, if present, can store the metadata in its own internal database. Some environments may include database systems in which metadata is to be stored.

## 2.2 Background

The following is an excerpt from the introduction section in the RDF specification that explains the value of metadata, and also the role played by the RDF format (which is described in more detail in Chapter 3, “XAP RDF Data Interchange Format”). While it presents a very Web-centric view of metadata, it also applies to other domains of usage as well, including print publishing.

> The World Wide Web was originally built for human consumption, and although everything on it is machine-readable, this data is not machine-understandable. It is very hard to automate anything on the Web, and because of the volume of information the Web contains, it is not possible to manage it manually. The solution proposed here is to use metadata to describe the data contained on the Web. Metadata is “data about data;” (for example, a library catalog is metadata, since it describes publications) or specifically in the context of this specification “data describing Web resources.” The distinction between “data” and “metadata;” is not an absolute one; it is a distinction created primarily by a particular application, and many times the same resource will be interpreted in both ways simultaneously. Resource Description Framework (RDF) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources. RDF can be used in a variety of application areas; for example: in resource discovery to provide better search engine capabilities, in cataloging for describing the content and content relationships available at a particular Web site, page, or digital library, by intelligent software agents to facilitate knowledge sharing and exchange, in content rating, in describing collections of pages that represent a single logical “document”, for describing intellectual property rights of Web pages, and for expressing the privacy preferences of a user as well as the privacy policies of a Web site.

## 2.3 Scope of XAP Metadata Framework

The XAP Metadata Framework consists of the following components:

- A XAP Metadata Model (Chapter 2).
- An XML-based interchange representation, based on the data model of the Internet standard RDF, for metadata (Chapter 3, "XAP RDF Data Interchange Format").
- A set of *schemas* for XAP-defined metadata (Chapter 4, "XAP Metadata Schemas").
- A set of conventions and rules for using the XAP-defined metadata. (See Chapter 5, "Metadata Property Commentary" and Chapter 7, "Application Integration Guidelines.")
- A set of conventions and rules for extending the metadata schema beyond what is defined by the XAP schemas (Chapter 6, "XAP Extensibility").

There are a number of areas, listed below, that are outside the scope of XAP, and should be under the control of the applications and tools that support XAP metadata. In some of these cases, even though the item is outside the scope, some recommendations are made in this document. In those cases, a reference to that information follows the bullet item in the list below.

The following are outside the scope of XAP:

- What metadata each specific application sets (XAP recommendations: Chapter 2, "XAP Metadata Framework," and Chapter 7, "Application Integration Guidelines").
- The user interface to metadata, if any.
- The operation of any media management systems (XAP recommendations: see Chapter 7, "Application Integration Guidelines").
- Which schemas beyond those defined by XAP are defined and present.
- Validity and consistency checking on metadata properties.
- Requiring that users set or edit metadata.

XAP schemas define a set of possible metadata. Not all XAP-defined metadata will be relevant to all documents and it is expected that only the relevant subset of metadata properties would be present in a particular document. Some XAP properties are defined to provide a standard method to store certain kinds of metadata. If that metadata is defined and relevant, the XAP-defined property should be used to store it.

Following the XAP schema and guidelines presented in this document cannot guarantee the integrity of metadata or metadata flow; that integrity must be accomplished and maintained by a specific set of applications and tools. An application's support for XAP refers to its ability to preserve and generate XAP format metadata, to give the user access to the metadata, and to support extension capabilities.

## 2.4 Model and Terminology

This section introduces the model and concepts that underlie the Adobe XAP Metadata Framework.

*Metadata* is information about the data that is contained in a file. The metadata may include information that is redundant with the content, that is a direct function of the content, and that is independent of the content. From a practical standpoint, defining metadata that is redundant with document content is avoided unless that metadata is of general interest to a variety of users and it is constant in size (that is, it doesn't grow in size with the document size). Metadata that is redundant with, or a function of document content, is called *internal metadata*. Metadata that supplements document content is called *external metadata*.

A *document* is any media object that a user might consider such as an image, text document, aggregate document (containing multiple text and image components), audio, video, etc. A document can be thought of as an abstraction. The title may change; the file in which it is stored may be renamed; one version of it may be in a database, another stored online; it may be rendered at various resolution to screen or various printers; a copy may be sent to someone by e-mail; and so on, but it remains the same document.

An *aggregate document* is one that has been constructed by incorporating other documents possibly with additional content into a single aggregate document.

*Document-level metadata* is information about a document as a whole. *Fine-grained metadata* is information about individual small elements of a document such as a paragraph or short sequence of words. XAP is concerned primarily with document-level metadata. Some of the properties defined by XAP can be used for fine-grained metadata, but the overall structure defined by XAP is not really appropriate for fine-gained metadata.

A *metadata schema* is a set of specific metadata property definitions.

A *version* of a document is the value of that document at some point in time. Versions are explicitly identified by being written to a filestore and being flagged as a new version of some previous version of a document. Each version is distinguished by having a version identifier. Change history information is associated with a version that indicates changes from its predecessor version.

The XAP model supports multiple versions of documents and multiple renditions of each version of a document.

A *rendition* of a document is a variation of the document derived from a particular version by changing the format or content in some well defined way. Common examples include the creation of a thumbnail or the generation of the same image in a different format or resolution. Each rendition is distinguished by having a name that is called the *rendition class name*. There may be many different renditions. Each kind of rendition has a different rendition class name, for example, *thumbnail*, *low-res*, or *French*.

A *resource* is the object that stores a particular version and rendition of a document and with which a set of metadata is associated. You can think of resources as the storage container (usually a file) for versions and renditions of documents. Resources could also store other kinds of objects.

The XAP schema defines *metadata* that is about a resource. The metadata consists of a number of properties; each *property* is associated with a resource and makes some statement about it. The statement has a *property name* and a *value* and has the form "the <property name> of

<resource> is <value>.” For example, the “author” of a book titled *The Programmer’s New Age Cookbook* might be “John Doe.”

A *value* can be a simple value:

- a boolean value (*TRUE* or *FALSE*)
- a text string
- date
- integer
- a real number (with an optional binary representation specified)
- a value chosen from a vocabulary of possible values (a *choice*)
- a value chosen from one of several vocabularies (an extensible *choice* or *xchoice*)

or it can be a structured value:

- an ordered sequence (*seq*) of values
- an unordered sequence (*bag*) of values
- a set of alternative values (*alt*)
- a nested structure with named fields each of which is itself a property.

Most simple values end up being text. There may be conventions or restrictions on what values are legal for a particular property.

Text strings and choice values can include a *vocabulary qualifier* that names a vocabulary from which the string is chosen.

A *vocabulary* is a set of possible values with some (informal) information about their semantics. A property value can be restricted to have values only from a vocabulary (or set of vocabularies). This is called a *closed vocabulary*. The property may also allow other values in addition to those listed in a vocabulary. This is called an *open vocabulary*. (See Section 4.4 for more information on XAP vocabularies.)

A sequence of values (*seq*) is simply a list of zero or more values. The order of the list has significance. A *bag* of values is similar to a sequence but the order does not have significance. A set of tagged alternative values (*alt*) is a list of alternative values where one can be chosen based on some criterion. The most common criterion is language.

The properties are grouped into *schemas*, each of which consists of a set of properties, a schema name, and a schema namespace prefix. The *schema namespace prefix* is a short abbreviation for the full schema name. The *schema name* serves to uniquely identify the schema, and, although it looks like a URL, it is simply a unique string.

Properties in the same or different schemas can be *aliased*, which means that they represent the same value. Aliased properties imply that they should always have the same value. The types of the two aliased properties must be the same, and changing one value means that all aliased values should be changed. The alias relationship is transitive. The reference to the aliased property uses XPath syntax. (see http://www.w3.org/TR/xpath)

Schema properties are categorized to help ensure that processing gives predictable results. The categories are:

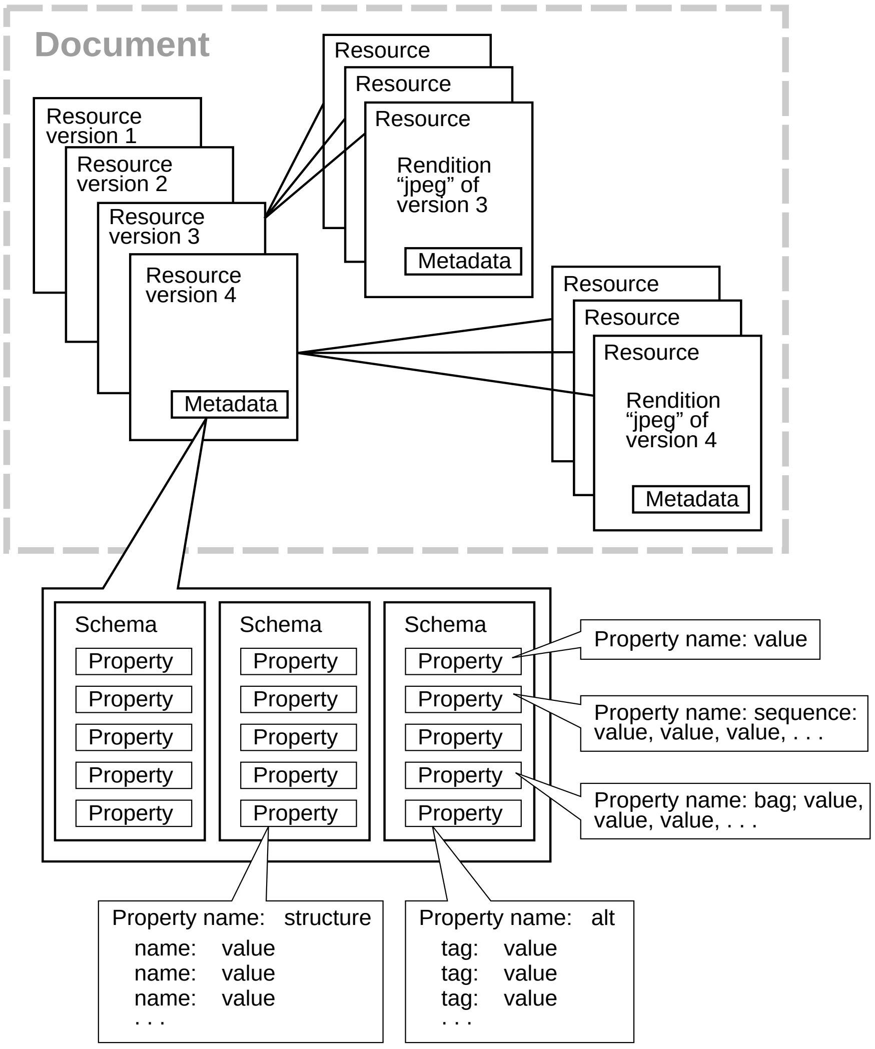
- *Internal*: metadata which is a reflection of the information contained within the resource. Modifications to internal properties are ignored by the authoring application. An example would be xapG:ColorSpace.
- *External*: metadata consisting largely of annotation information. Modifications may be displayed by the authoring application but are not acted upon. An example would be xap:CreateDate.
- *Relational*: a subset of internal metadata that pertains to the relationship of this resource with outside systems. Examples might include management markers or print resolution. Modifications to these properties outside the authoring application should be resolved against the internal information in the document.

Properties from a schema typically have values, or they may instead be absent in the metadata of a given resource. Most properties are initially absent until they are given a value for the first time. They may also be deleted. The presence or absence of a property is visible to application programs. Every present RDF property must have a value, even if it is just the empty string.

XAP metadata can be stored within a resource, in a separate resource related by convention to a resource, and/or in a media management system.

Figure 2.1 shows the relationship between a document and the components of the metadata. It reflects the model that a document is an abstraction, and the implementation of it may consist of multiple renditions, versions, and renderings, represented by a number of resources or files, each containing metadata. It does not attempt to illustrate how the metadata or versions are managed (see Chapter 7, “Application Integration Guidelines, for more information on that topic).

***FIGURE 2.1 Documents and Metadata Component Relationships***



In summary:

- XAP metadata is associated with a resource, which stores a particular version and rendition of a document.
- XAP metadata is organized into schemas each of which has a schema name, a schema namespace prefix and a set of properties.
- Each property has a property name and value and says something about the resource it is associated with.

- Property values can be text, dates, booleans, integer or real numbers, values chosen from one or more vocabularies, sequences of values, bags of values, a set of alternative values, or nested structure containing more properties.

## 2.5 Granularity of XAP Metadata Associations

XAP metadata can be associated with any structure or substructure from servers and file systems, to files and database entries, to pages, lines, characters, styles and other finer grained structures stored within a file. Keep in mind the practical issues which will argue against using XAP for very find-grained metadata:

- XAP works best if the document containing the metadata has an identity. Assigning and tracking identities for individual words and characters is largely impractical.
- XAP includes a certain amount of overhead given its XML representation (tag names, angle brackets, timestamps, etc.). Thus, storage overhead can be considerable if XAP is associated with very small structures.

# 3 XAP RDF Data Interchange Format

## 3.1 Introduction

This chapter describes the Web-standard technology Resource Description Framework (RDFMS 1.0) which is used as the data interchange format. This chapter also includes a specification of XML Packets, which describes how applications can embed metadata packets in files so they can be located by simple scanning rather than by parsing.

The Adobe XAP Metadata Framework uses a profile (subset) of the RDF specification developed by W3C. Use of unsupported features of RDF may result in incorrect operation of XAP-enabled applications. See Section 3.5, "RDF Features Not Supported in XAP" for more details.

**NOTE:** This chapter assumes that you are familiar with basic RDF concepts. RDF terminology which is used only in this chapter, is not defined here or in the more general XAP Framework discussion in the previous chapter.

## 3.2 Background

A goal of XAP was to establish a standard for the representation and interchange of metadata between applications. Rather than invent a new standard, it made sense to adopt the RDF standard, and benefit from the documentation, tools, and shared implementation experience that come with an open Internet standard. The RDF syntax is based on XML.

Most documentation about RDF uses the term *resource* as the thing that is being described by the metadata. *Resource* is used here in the same sense as it is used on the Web: it is the abstraction that contains information; it may or may not be a file.

## 3.3 RDF Data Model

The following sections explain the RDF data model, and how XAP uses the RDF model to support its goals.

There are three basic object types: Resources, such as documents and media files or literal values, Properties, such as Title and Author, and Statements, which relate Properties to Resources. All Statements in XAP can be expressed as triples: {Subject, Predicate, Object}. For simple name/value pairs, the Predicate is the name, the Object is the value, and the Subject is the document or media file that the name/value pair describes. RDF represents these relationships as directed graphs, where the Subject and Object are nodes, and the Predicate is a named directed arc from the Subject to the Object.

Consider as a simple example the sentence:

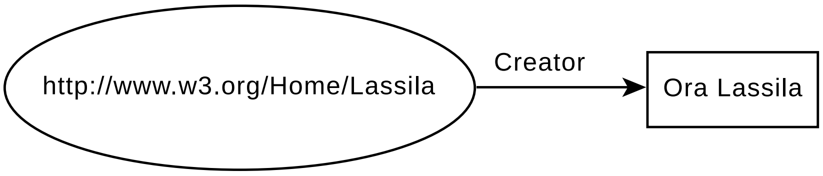
*Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.*

This sentence has the following parts:

| | |
|---|---|
| Subject (Resource): | http://www.w3.org/Home/Lassila |
| Predicate (Property): | Creator |
| Object (literal): | Ora Lassila |

In RDF diagrams, the nodes (drawn as ovals) represent resources and arcs represent named properties. Nodes that represent string literals will be drawn as rectangles. The sentence above would thus be diagrammed as shown in Figure 3.1.

***FIGURE 3.1 Simple node and arc diagram***



## 3.4 How XAP Uses the RDF Data Model

This section describes how XAP uses the RDF data model. It explains the basic model; explains how schemas are named and includes sample code that illustrates how to do it; and shows how to specify the locale for a document. The issues of extensibility and aliasing are also explained.

### 3.4.1 Description Object

A Description object is an RDF element that contains statements about a resource. Properties from various schemas are attached to the Description object. In other words, if you took all the statements in a model, and factored them by common subjects (all about the same resource), you would have a set of top-level Description objects, one for each unique subject resource.

Metadata can be associated with resources either internally or externally. *Internal association* usually means that the actual Description object is embedded in the data of the resource somewhere. *External association* usually means that the Description object refers to the resource with some kind of pointer, such as a URL. External associations can also be managed by a database application, which can maintain statement triples directly. All of these methods are supported by this data model.

### 3.4.2 Multiple Values

The XAP subset of the RDF data model includes the container mechanism (described in section 3 of the RDF specification) for modeling properties with multiple values. The *syntax* also allows properties to be repeated, but this practice is not allowed for XAP in order to avoid ambiguity between the two cases of a single property with multiple values, and multiple instances of the same property with single values. Avoiding multiple values also makes it easier to implement XAP for protocols like WebDAV (an IETF Proposed Standard, published as RFC 2518) which forbid repeating properties. Repeated properties in the syntax are mapped into an equivalent structured container type, which is described by the schema. When no schema description is available, the default is a sequence type.

### 3.4.3 Schemas and Name Spaces

Metadata can usually be organized into related groups of items. These groups are relevant only for particular types of documents, or perhaps only for certain stages of a workflow. These groups are implemented by defining schemas for them. These schemas create a new name space.

A schema is a set of specific metadata property definitions; it includes, but is not limited to:

- A vocabulary of elements (property names)
- The legal values for elements (constraints on property values)
- The relationship between properties, if any (there’s none if they are simple and flat)

The vocabulary of properties is defined within a *name space*. A name space helps to differentiate between predicate names that are the same, but which have different meanings depending on the schema. For example, in one schema, *Creator* may mean the person who created a resource. In another schema, *Creator* may mean the application used to create a resource. We want to use the same value, but we don’t want programs to become confused. By specifying that each name must be defined within a name space, there is no confusion.

In XAP metadata, each schema is named. The URI of the schema is considered to be the name of the name space.

Some properties have values that contain more than one component (structured values). The names of the components use tags from an XML namespace associated with the structure. An example of such a property would be the xapG:NaturalDimensions property. Each property requiring specification of dimensions needs a width, a height, and an indication of the units used for the numeric values of width and height. Each property describes a different interpretation of dimension: physical dimensions, pixel sample dimensions, desired rendering dimensions, etc., and all use tags from their own namespace. This namespace is just a simple factoring of the vocabulary into useful, independent modules. This is more for the convenience of programmers and human readers of metadata specifications than for any other reason.

Let’s suppose we have an image resource “myPhoto.gif.” We want to describe the dimensions in pixels, as well as the natural presentation dimensions (desired size in inches when displayed at 100% scale). These are two different properties which share the same value type.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description about="myPhoto.gif"
      xmlns:xapG="http://ns.adobe.com/xap/1.0/g/"
      xmlns:stDims="http://ns.adobe.com/xap/1.0/sType/Dimensions#">
        <xapG:NaturalDimensions rdf:parseType='Resource'>
          <stDims:w>4</stDims:w>
          <stDims:h>3</stDims:h>
          <stDims:unit>inches</stDims:unit>
        </xapG:NaturalDimensions>
    </rdf:Description>

    <rdf:Description about="myPhoto.gif"
      xmlns:xapGImg="http://ns.adobe.com/xap/1.0/g/img/"
      xmlns:stDims="http://ns.adobe.com/xap/1.0/sType/Dimensions#">
        <xapGImg:Dimensions rdf:parseType='Resource'>
          <stDims:w>640</stDims:w>
          <stDims:h>480</stDims:h>
          <stDims:unit>pixels</stDims:unit>
        </xapGImg:Dimensions>
    </rdf:Description>
</rdf:RDF>
```

The first description uses the schema for XAP Graphics (http://ns.adobe.com/xap/1.0/g/). The property is called xapG:NaturalDimensions, and its value is a nested description (which is what `rdf:parseType='Resource'` means). The nested description uses the namespace “http://ns.adobe.com/xap/1.0/sType/Dimensions#.”

The second description uses the schema for XAP Graphics: Image:

```
(http://ns.adobe.com/xap/1.0/g/img/)
```

The property is called xapGImg:Dimensions, and its value is also a nested description. The nested description also uses the namespace http://ns.adobe.com/xap/1.0/sType/Dimensions#.

### 3.4.4 Locale Support

The data model supports localization by qualifying Objects (values) with a language attribute. For example, you can specify that a Title is in French, while the Keywords are in English. However, for each literal Object value, only one such attribute is allowed. If multiple locales are required for a single Property, the alternation container type allows a single property to have different values for different locales. The data model provides Object values with their associated locale, in this case.

Because RDF is based on XML, localization is relatively easy. First, the default character encoding for XML is UTF-8. If you need two-byte characters, you can encode XML in any two-byte Unicode encoding you like, such as UTF-16.

Furthermore, through the `xml:lang` attribute on metadata values, you can localize metadata values on an item by item basis. For example, you can have a Title in French, and a Description in English, or a Price in German Deutschmarks and a Weight in metric tons. The alternation mechanism for values even allows you to specify a preferred language, and several alternative languages, all for the same property.

Again, we have to use the basic syntax in order to apply the xml:lang attributes to individual values, and we use Dublin Core which defines alternation by language for the title. Suppose we wanted to indicate that the preferred title of this document is in English, but there is an alternative title for French and German:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description about="http://atg.corp.adobe.com/projects/xap/xap.html"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:creator>
        <rdf:Bag>
          <rdf:li>John Doe</rdf:li>
          <rdf:li>Jane Smith</rdf:li>
        </rdf:Bag>
      </dc:creator>
      <dc:title>
        <rdf:Alt>
          <rdf:li xml:lang="en">
            Adobe XAP Metadata Framework
          </rdf:li>
          <rdf:li xml:lang="fr">
            Adobe XAP Structure du Metadata
          </rdf:li>
          <rdf:li xml:lang="de">
           Rahmenbedingungen für Adobe XAP Metadaten
          </rdf:li>
       </dc:title>
    </rdf:Description>
</rdf:RDF>
```

### 3.4.5 Extensibility

XAP Metadata schemas may be added by the procedure described in *Chapter 6, "XAP Extensibility*," which also shows examples of extensions.

The data model makes no distinction between standard and non-standard schemas. All properties are associated with a namespace URI, and that URI is the name of the schema.

The version of a schema is considered to be a part of the schema name. Therefore, *foo:/schema/1.0/* and *foo:/schema/2.0/* are considered to be two completely different schemas with no relationship between them. This leaves applications free to define the semantics for schema versions.

The data model does not require the presence of a schema in order to instantiate and explore statements.

### 3.4.6 Aliasing

One of the goals of the XAP framework is to efficiently handle properties from different schemas which have the same logical meaning. This goal is achieved through the aliasing of properties.

Two properties should be aliased if and only if they have the same value type and meaning. There must not be loops in the alias definitions. For example, the following must not occur: *A* is aliased to *B* aliased to *C* aliased to *A*.

## 3.5 RDF Features Not Supported in XAP

The following features of the RDFMS 1.0 specification are *not* supported by XAP:

- The optional `rdf:RDF` element (`rdf:RDF` is required)
- Top-level containers (top-level must be `rdf:Description` or `typedNode` elements)
- The `rdf:ID` attribute (ignored on all `rdf:Description` or `propertyElt` elements)
- The `rdf:bagID` attribute (ignored)
- The `rdf:aboutEach` or `rdf:aboutEachPrefix` attributes (entire `rdf:Description` ignored)
- Reified statements (they are not generated in the model).

Furthermore, all `rdf:about` attributes must appear on top-level `rdf:Description` or top-level `typedNode` elements only. The `rdf:about` values must all be the same within a single `rdf:RDF` element. In general, the value of rdf:about is the empty string “”, which implies that the XAP metadata is embedded in the document that it describes. The value of `rdf:about` may be a URI, in which case the metadata describes the document identified by the URI, and is not embedded in that document (though it may be embedded in a different document).

Otherwise, all other features of RDFMS 1.0 are supported in XAP.

## 3.6 Representation and Storage of Metadata

XAP metadata uses the XML serialization of RDF as the standard for metadata interchange. In addition, the XML data can be embedded into any kind of document using the XAP Packet format described in Section 3.8, “XAP Packets.”

### 3.6.1 XML Representation Examples

The following examples of XAP representation are written in XML, which should be familiar to anyone who has done any hand-coding of HTML.

### 3.6.1.1 XAP Examples in RDF

Familiarity with XML and RDF would be helpful, but even if you are not familiar with those standards, you can get the flavor for how XAP would use RDF through these examples. It is also helpful to know about XML name spaces.

One hint that might help: the XML name space mechanism is invoked by defining an attribute called “xmlns.” The format is:

```
xmlns:foo="bar"
```

This means that any element or attribute that begins with the prefix “foo:” is part of the “bar” name space. So, for example:

```
foo:Author="Bubba"
```

means the element or attribute *Author* is in the name space called *bar*. The *foo* part can be any legal XML name (without colons), and we usually pick a short prefix since it could be used a lot. If two elements have the same name, but different name space prefixes, they are considered to be different elements in XML:

```
<EXAMPLE xmlns:foo="bar" xmlns:oof="rab">
    <DIFFERENT foo:Author="Bubba" oof:Author="Not Bubba"/>
</EXAMPLE>
```

### 3.6.1.2 Simple Example of XAP Metadata in XML

Suppose we want to associate some Acrobat metadata with this document, the *Adobe XAP Metadata Framework*. We want to say that the author is John Doe, and that the title is “Adobe XAP Metadata Framework.” We use an RDF *Description* to group all Statements about the same resource together. Here is the RDF syntax:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
     <rdf:Description about="http://atg.corp.adobe.com/projects/xap/xap.html"
       xmlns:pdf="http://na.adobe.com/pdf/1.3/">
          <pdf:Author>John Doe</pdf:Author>
          <pdf:Title>Adobe XAP Metadata Framework</pdf:Title>
    </rdf:Description>
</rdf:RDF>
```

The above code uses the basic syntax above, which is needed when you have complex metadata to express. However, in the simple case above, which is just two flat name/value pairs, there is an alternative syntax which is more concise:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <rdf:Description about="" xmlns:pdf="http://ns.adobe.com/pdf/1.3/"
         pdf:Author="John Doe"
         pdf:Title="Adobe XAP Metadata Framework"/>
</rdf:RDF>
```

According to the XAP data model, each property belongs to a schema. This is done using the xmlns attribute in the Description element. The URI for the Adobe PDF Schema is *http://www.adobe.com/std/ns/pdf/1.3/*.

Let’s say we wanted to add some XAP core metadata, such as the MIME type of this document (xap:Format), and the *language* it is in (xap:Locale).

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description about="" xmlns:pdf="http://ns.adobe.com/pdf/1.3/"
        pdf:Author="John Doe"
        pdf:Title="Adobe XAP Metadata Framework"/>

    <rdf:Description about="" xmlns:xap="http://ns.adobe.com/xap/1.0/"
        xap:Format="text/html" xap:Locale="en"/>
</rdf:RDF>
```

The formal name of the core XAP metadata schema is:

```
http://ns.adobe.com/xap/1.0/"
```

If you want to specify a co-author, we need to use a list:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description about="http://docs.adobe.com/projects/xap/xap.html"
      xmlns:dc="http://purl.org/dc/elements/1.1/">
         <dc:creator>
           <rdf:Bag>
             <rdf:li>John Doe</rdf:li>
             <rdf:li>Jane Smith</rdf:li>
           </rdf:Bag>
         </dc:creator>
     </rdf:Description>
</rdf:RDF>
```

The `rdf:li` elements are meant to remind you of HTML <li> list items. An `rdf:Bag` is the name of the unordered collection container type. The important point here is that multiple values can be specified for a property, and we can say something about how those values are structured.

## 3.7 Limitations of RDF

RDF is not well suited for, or was not designed to handle, the following features:

- Inline binary objects (primarily due to XML syntax).
- Optimal (minimal) size of the representation.
- Knowledge representation, in the most general artificial intelligence sense.
- Validation against a DTD.

## 3.8 XML Packets

The XML Packet format was developed to enable simple scanners to find XML data embedded in files with formats that a simple scanner may not understand, such as Photoshop® or PDF files. The format uses a syntax that is as close to XML as possible to minimize the filtering burden on the simple scanner.

The XML Packet format was designed to accomplish the following:

- Support embedding in binary and text formats, including the various Unicode encodings, but excluding XML itself (which has its own rules for embedding).
- Deal with arbitrary positioning within a byte stream (so as not to rely on machine word boundaries, etc.).
- Enable multiple XML packets to be embedded in a single data file.
- Provide easy-to-scan markers for delimiting the XML packet. Such markers should be XML syntax compatible to allow transmission to an XML parser without additional filtering.
- Enable in-place editing, including expansion, of embedded XML packets.

**NOTE:** Any XML packet containing one or more XAP metadata elements will be referred to in this document as a *XAP Packet*.

The procedure for creating a XAP Packet is described in this section. The packet includes a header and trailer (see Figure 3.3). The header provides byte ordering information, and optional byte length and encoding information.

**NOTE:** Be aware that XAP Packets can contain non-XAP data; be careful when writing the data to make sure that non-XAP data is not corrupted or omitted.

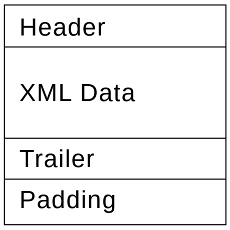
Here is an example of packetized XML data. The XML data uses UTF-16 (16-bit per character) encoding in big-endian order:

```
<?xpacket begin='■' id='W5M0MpCehiHzreSzNTczkc9d' bytes='1496' encoding='UTF-
16'?>
 ... 700 bytes of XML data text ...
<?xpacket end='w'?>
 ... 500 bytes of padding ...
```

Where '■' represents "U+FEFF."

Figure 3.3 shows a schematic of an XAP Packet.

**FIGURE 3.3 XAP Packet Schematic**



The entire packet must conform to the Well-Formedness requirements of the XML specification. Also, there are additional constraints:

- A single file *may* contain zero, one, or multiple packets.
- Different packets *may* be in different character encodings.
- Packets *must not* nest.
- Data *attributes* in processing instructions are separated by exactly one blank U+0020 character.

The following sections describe the parts of the packet illustrated in Figure 3.3.

### Header

The Header is an XML processing instruction:

<?xpacket ... ?>

The remainder of the processing instruction contains information about the packet. The syntax observes XML attribute syntax, which is production [41] Attribute, which is roughly:

```
Attribute ::= Name '=' AttValue
AttValue ::= '"' ([^<&"] | Reference)* '"'
     | "'" ([^<&'] | Reference)* "'"
```

Note the use of either matching single or double quotes. Otherwise, a common error would be the use of the wrong quote character.

The description of each attribute follows.

#### *Attribute: begin*

This mandatory attribute is present only in the initial processing instruction, and indicates that it is the beginning of a new packet. The value of this attribute is either the Unicode BOM (Byte Order Marker) character U+FEFF or nothing (‘’ or “”). In any 16- or 32-bit Unicode encoding, the U+FEFF character serves as a BOM, where the character is written in the natural order of the authoring/generating application (consistent with the byte order of the XML data encoding).

As described in the *Usage Hints* below, an XAP Packet processor should be reading a single byte at a time until it has successfully interpreted a valid packet header. While processing the value of the id attribute, if the processor detects the byte value ‘0xFE’ followed by ‘0xFF,’ it knows that the packet is big-endian order. If the processor detects the byte value ‘0xFF’ followed by ‘0xFE,’ it knows that the packet is little-endian order. If the processor detects the byte value ‘0xEF,’ followed by ‘0xBB,’ followed by ‘0xBF,’ it knows this is UTF-8. If the attribute has no value (quote or double quote followed immediate by another quote or double quote), the byte order is irrelevant and the overall character encoding *must not* be any 16 or 32-bit Unicode encoding (that is, UTF-8, US-ASCII, etc.).

*Attribute: id*

Next, there is a mandatory id. For all packets defined by this version of the syntax, the value of the id is the following string of 7-bit ASCII characters:

```
W5M0MpCehiHzreSzNTczkc9d
```

The value of the attribute *must* be encoded in the character encoding of the overall packet (see below). Thus, if the overall encoding is big-endian UTF-16, the id value should be converted from 7-bit ASCII to UTF-16.

*Attribute: bytes*

The id may be followed by an optional byte length attribute. The value of this attribute is the number of bytes in the entire packet, including *Header*, *Trailer*, and *Padding*.

*Attribute: encoding*

The id or bytes attribute may be followed by an optional encoding attribute. It is identical to the encoding attribute in the XML declaration (see productions [23] and [80] in the XML specification). It specifies the character encoding of the entire packet. If this attribute is omitted, the encoding of the packet *must be* UTF-8.

### XML Data

The bytes of the XML data are placed here. If the encoding is specified in the Header, the encoding of the XML data must match. If the encoding was omitted from the Header, the encoding of the XML data must be UTF-8.

You *must* omit the XML declaration for the XML data when using this packet syntax for embedding. The XML specification requires that the XML declaration be “the first thing in the entity.” This will never be the case for an embedded XAP Packet, the somewhat ambiguous definition of “entity” with respect to embedding notwithstanding. You may preserve the information contained in your XML declaration by translating it into a comment or a processing instruction, such as:

```
<;?was-xml version="1.0" standalone="yes"?>
```

#### Trailer

This mandatory processing instruction indicates the end of the XML data and the beginning of well-formed padding, if any.

```
<?xpacket ... ?>
```

This processing instruction has one attribute, described as follows:

*Attribute: end*

This mandatory attribute indicates that this is the trailer. The value of the attribute is either “r” or “w.” If “r,” the packet is “read-only” and should not be updated in-place. If “w,” the packet may be updated in-place if and only if there is available space through the padding. If the size of the Header+XML data+Trailer is less than it was before the update, the padding should be increased accordingly so that the overall packet size remains constant. Use the value “r” for file formats which compute invariants over all of their contents, such as checksums. If in doubt, use “r.”

#### Padding

In order to enable in-place edits and expansion of the embedded XML, padding should be added to the end of the packet so that additions and edits may be easily made to the packet without overwriting existing application data.. It is highly recommended that applications allocate 50% of the packet size as padding, with a minimum of 4 KB. This padding must conform to the XML Well-Formedness requirements, which requires that it must be XML compatible whitespace, comments, or processing instructions. The recommended practice is to use the blank character (U+0020) for padding, in the appropriate encoding.

### 3.8.1 Usage Hints

A file should be scanned byte-by-byte until a valid header is found.

A simple scanner should begin its scanning state machine by looking for one of the following byte patterns (which represents “<?xpacket begin=”):

16-bit encoding (UCS-2, UTF-16): (either order)

```
0x3C 0x00 0x3F 0x00 0x78 0x00 0x70
0x00 0x61 0x00 0x63 0x00 0x6B 0x00 0x65 0x00 0x74 0x00 0x20 0x00
0x62 0x00 0x65 0x00 0x67 0x00 0x69 0x00 0x6E 0x00 0x3D [0x00]
```

8-bit or multibyte encoding (UTF-8, ASCII 7-bit, ISOLatin-1):

```
0x3C 0x3F 0x78 0x70 0x61 0x63 0x6B
0x65 0x74 0x20 0x62 0x65 0x67 0x69 0x6E 0x3D
```

The 32-bit UCS-4 pattern is similar to the UCS-2 pattern above, only with three 0x00 bytes for every one in the UCS-2 version.

For 16-bit encodings, a simple scanner cannot be sure whether the 0x00 values are in the high or low order half of the character until you read the byte order mark (the value of the begin attribute). As you can see from the pattern, it starts with the first non-zero value, regardless of byte order, which means that there may or may not be a terminal 0x00 value.

A simple scanner may choose to simply skip 0x00 values and search for the 8-bit pattern. Once the byte order is established, the scanner should switch to consuming characters rather than bytes. After finding a matching byte pattern, the scanner must consume a quote or double-quote character. The scanner is now ready to read the value of the begin attribute.

| | |
|---|---|
| 16-bit encoding, big endian: | `0xFE 0xFF` |
| 16-bit encoding, little endian: | `0xFF 0xFE` |
| UTF-8: | `0xEF 0xBB 0xBF` |

Then the scanner consumes the closing quote or double-quote character. The scanner now has enough information to process the rest of the header in the appropriate character encoding. This is a potential packet. Once the id is matched, it can be assumed that it is a valid packet.

*Single or Double quote*

Remember that the attribute values in the processing instruction may have either single or double quotes. The following header is well-formed:

```
<?xpacket begin="" id='W5M0MpCehiHzreSzNTczkc9d' encoding="UTF-8"?>
```

**NOTE:** Do not use the BOM outside of the packet header!

The character U+FEFF is used in UTF-16 encoding to hint at byte order. Since this information is already represented in the packet header, this Binary Order Marker (BOM) is unnecessary. Furthermore, well-formedness requirements constrain the BOM to be the first two bytes of the document entity, which is impossible in this syntax.

If your XML data has a BOM, remove it before embedding the XML data in the packet.

# 4 XAP Metadata Schemas

## 4.1 The Adobe XAP Schemas

This chapter defines the Adobe Standard Metadata Schema, and specifies the property value types for each schema and the associated schema vocabularies which list the allowed values. The sections include:

- Section 4.2, “Adobe Standard XAP Metadata Schema”
- Section 4.3, “Property Value Types”
- Section 4.4, “XAP Vocabularies”

In addition, Chapter 5, “Metadata Property Commentary” describes the use of, and rationale for XAP metadata properties, and the appendix specifies the PDF and Dublin Core Schemas.

### 4.1.1 Property Value Type Representation

In the *Value Type* column of each schema table, a specific notation is used. A type preceded by a modifier, which is derived from RDF nomenclature, is defined as follows:

| | |
|---|---|
| *alt* | An alternation. The first value in the sequence is the preferred value; all subsequent values are alternatives. |
| *bag* | An unordered collection of values of the specified type. |
| *seq* | An ordered sequence of values of the specified type. |

Most of the Value Types are self-explanatory, but some need additional explanation, or are in fact references to structures. All value types are defined in Section 4.3, “Property Value Types.” For an explanation of the Category column, see “Property Categories” on page 62.

For properties whose values are strings, for which the Lang value is unknown, the following notation is used to indicate the default value to be used (for example, see the entry for pdf:Title in the PDF Schema in the appendix):

| **Property Reference:** | **Interpretation:** |
|---|---|
| xap:title/*[@xml:lang=‘x-default’] | alternation by language; selects ‘x-default’ |

## 4.2 Adobe Standard XAP Metadata Schema

Each XAP schema described in this chapter is shown in table form, along with the namespace string that identifies the schema. Each table lists all properties defined for that schema as well as the value type, category, and whether the property is aliased to any other XAP property. Additional information and commentary is given for most properties in Chapter 5, "Metadata Property Commentary."

**NOTE:** If you are viewing this document online, *Property* column entries in the following tables are linked to the corresponding entry in the Metadata Property Commentary table in the next chapter. Also, Value Type entries are linked to the corresponding table entry in Section 4.3, "Property Value Types."

## 4.2.1 XAP Core Schema

The XAP Core Schema contains metadata properties that are common to all applications.

***TABLE 4.1 XAP Core Schema***

*The namespace prefix is xap. The schema name is http://ns.adobe.com/xap/1.0/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xap:Advisory | bag XPath | List of properties edited outside of the authoring application. | External |
| xap:Author | ProperName | Author of the document | External |
| xap:BaseURL | URL | base URL for relative URLs | Relational |
| xap:CreateDate | Date | Document creation time. | Internal |
| xap:CreatorTool | AgentName | Tool that created the resource. | Internal |
| xap:Description | alt Text | A textual description of the content | External |
| xap:Format | MIMEType | Distinguish various resource formats | Internal |
| xap:Keywords | bag Text | List of descriptive phrases | External |
| xap:Locale | Locale | Default Locale | Internal |
| xap:MetadataDate | Date | Last metadata modification time | Internal |
| xap:ModifyDate | Date | Last resource modify time | Internal |
| xap:Nickname | Text | Short informal name for resource | External |
| xap:Rights | alt Text | Inline text of copyright or other rights statements. ***Note:*** Use of this property is deprecated; use xapRights:Copyright in its place. | External |
| xap:Title | alt Text | Document Title | External |

### 4.2.2 XAP Media-Type Schemas

***TABLE 4.2 XAP Graphics Schema***

*The namespace prefix is xapG. The namespace is http://ns.adobe.com/xap/1.0/g/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapG:ColorSpace | ColorMode | The name of the group of colors from which colors in the document are taken. | Internal |
| xapG:Compression | XChoice (open) | The name of the compression algorithm used to compress all or part of the document. | Internal |
| xapG:GraphicsType | XChoice (closed) | Raster, Vector, Dynamic | Internal |
| xapG:NaturalDimensions | Dimensions | Presentation dimensions | Relational |
| xapG:NumberOfColors | Integer | Number of colors in color space (256, 65536, etc.) | Internal |
| xapG:NumberOfInks | Integer | Number of process and spot colors needed to print entire document including any contained documents. | Internal |

***TABLE 4.3 XAP Graphics: Image Schema***

*The namespace prefix is xapGImg. The namespace is http://ns.adobe.com/xap/1.0/g/img/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapGImg:Dimensions | Dimensions | Image dimensions in sampling unit | Internal |
| xapGImg:Resolution | Resolution | Number of pixels per unit measure | Relational |

***TABLE 4.4 XAP Dynamic Media Schema***

*The namespace prefix is xapDyn. The namespace is http://ns.adobe.com/xap/1.0/dyn/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapDyn:Duration | Duration | Total duration of resource | Relational |
| xapDyn:NTracks | Integer | Number of tracks or channels | Internal |

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapDyn:Tracks | seq TrackDesc | Array of track descriptions | External |

**TABLE 4.5 XAP Dynamic Media: Video Schema**

*The namespace prefix is xapDynV. The namespace is http://ns.adobe.com/xap/1.0/dyn/v/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapDynV:BitRate | Integer | Bits per second | Relational |
| xapDynV:Dimensions | Dimensions | Of playback view rectangle | Relational |
| xapDynV:Interleaved | Boolean | If true, NTSC fields, otherwise frames | Internal |
| xapDynV:NaturalRate | Real | Fields/Frames per second | Relational |

**TABLE 4.6 XAP Dynamic Media: Audio Schema**

*The namespace prefix is xapDynA. The namespace is http://ns.adobe.com/xap/1.0/dyn/a/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapDynA:ChannelCount | Integer | Number of audio channels | Internal |
| xapDynA:Compression | XChoice (open) | Audio compression technique | Internal |
| xapDynA:Rate | Real | Samples per second | Relational |
| xapDynA:SampleSize | Integer | Number of bits per sample | Internal |
| xapDynA:Volume | Real | 0.0 = silence, 1.0 = maximum volume | Relational |

***TABLE 4.7 XAP Text Schema***

*The namespace prefix is xapT. The namespace is http://ns.adobe.com/xap/1.0/t/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapT:Encoding | TextEncoding | Specifies the name of the character encoding standard used for text in the document. For example, ISO-8859-1 | Internal |
| xapT:FontList | bag Font | Lists the names of all fonts used in the document. | Internal |

***TABLE 4.8 XAP Text: Paged-Text Schema***

*The namespace prefix is xapTPg. The namespace is http://ns.adobe.com/xap/1.0/t/pg/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapTPg:MaxPageSize | Dimensions | Size of the largest page in the document (including any in contained documents) | Internal |
| xapTPg:NPages | Integer | Number of pages in the document (including any in contained documents) | Internal |

### 4.2.3 XAP Media Management Schema

Resources are identified by a document identifier, a version identifier, and a rendition class name. All versions and renditions of a document share the same document identifier. The triple (document identifier, version identifier, rendition class name) forms a key which can be used to uniquely identify each media resource. A media management tool (or simple convention for that matter) can map the triple to a filename or URL.

The creating application should assign the DocumentID. Other tools and applications that operate on the document should preserve the assigned DocumentID.

If an application has media management plug-in interfaces, it may operate in conjunction with the media management system to assign the DocumentID.

**TABLE 4.9 *XAP Media Management Schema***

*The namespace prefix is* xapMM. *The namespace is* http://ns.adobe.com/xap/1.0/mm/.

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapMM:ContainedResources | bag ResourceRef | A bag referencing all resources that may be contained in this one. | Internal |
| xapMM:ContributorResources | bag ResourceRef | Resource that in some way contributed to this one, implying a dependency. | Internal |
| xapMM:DocumentID | URI | The common identifier for all versions and renditions of a document. | Internal |
| xapMM:History | seq ResourceEvent | A sequence of high-level user actions that resulted in this resource. | Internal |
| xapMM:LastURL | URL | Last place this resource was written. | Relational |
| xapMM:Manager | AgentName | Name of the software agent that manages this resource. | Relational |
| xapMM:ManageTo | URL | Location of managed rendition of asset. | Relational |
| xapMM:RenditionClass | RenditionClass | Rendition class name of this resource. | Internal |
| xapMM:RenditionOf | ResourceRef | Resource that this resource is a rendition of. | Internal |
| xapMM:SaveID | Integer | Incremented on each write to LastURL. | Relational |
| xapMM:VersionID | Text | The document version identifier for this resource. | Internal |
| xapMM:Versions | seq Version | Part of the version history of this document. | Internal |

Authoring applications should maintain all of the xapMM properties except Manager and Versions which are primarily for use by a media management system. By setting these properties, an audit trail, embedded in the document itself, is maintained that allows a follow-on media manager to reconstruct the relationships among documents found on a user's unmanaged storage system.

A media management system should preserve media management metadata unchanged and update the xapMM metadata to be consistent with the results of operations performed in the media management system.

#### 4.2.3.1 Versions

Creation of a new version usually requires some explicit action on the part of the user and there, consequently, must be some user interface that allows the application to deduce that a new version is to be created. This would commonly be the check-in operation for a media management system.

From the point of view of XAP metadata, all that is required to change the version number is to assign a new value for xapMM:VersionID, typically by incrementing it.

The media management system is responsible for storage of any old versions of the document. XAP is not involved in mapping resources to storage pathnames or in retention of old versions.

#### 4.2.3.2 Renditions

Renditions of a version of a document are created directly by some explicit user action or indirectly by some tool that is carrying out some user action. Renditions may be stored as normal resources or files, or may be embedded inside another resource by an aggregating application.

Creation of a rendition involves changing the xapMM:RenditionClass to a value that identifies the kind of rendition being created, and changing the xapMM:RenditionOf property to indicate the resource from which the new rendition is derived.

If an aggregating application creates a new rendition of a resource as part of embedding a resource, the metadata for the modified resource should be changed as indicated in the previous paragraph and the xapMM:ContainedResources property of the aggregate resource should be modified to include the new, embedded rendition.

### 4.2.4 XAP Support Schema

The following properties support cataloging and media management features. These properties do not describe a resource directly, as Author and Title do. Rather, they describe information that is related to the resource, such as the way it is identified in a storage management system.

These properties should never be embedded in the resource they describe. They are strictly for metadata managers that control metadata external to the resource file. It is expected that specific media management systems will define additional properties to support their individual features.

***TABLE 4.10 XAP Support Schema***

*The namespace prefix is xapS. The namespace is http://ns.adobe.com/xap/1.0/s/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapS:EntityTag | Text | Supports cache validation when comparing the “content value” of two blocks of metadata. Follows a convention based on the “entity” concept defined in HTTP/1.1. See http://www.ietf.org/rfc/rfc2616.txt | Relational |
| xapS:FileDisposition | alt FileDisposition | Preferred file name to save on disk, or preferred URL for publishing on a web server, by OS. | Relational |
| xapS:ResourceID | URI | The unique identifier for this particular resource. Used for storage systems which provide a unique identifier for a stored resource which is incompatible with the format defined in xapMM. | Relational |
| xapS:Size | Integer | File size in bytes | Relational |

## 4.2.5 XAP Basic Job Ticket Schema

The following schema describes very simple workflow or job information.

***TABLE 4.11 XAP Basic Job Ticket Schema***

*The namespace prefix is xapBJ. The namespace is http://ns.adobe.com/xap/1.0/bj/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapBJ:JobRef | bag Job | Jobs which this document is part of. | External |

### 4.2.6 XAP Rights Management Schema

This schema deals with rights management. All properties are optional. All alternations are for multiple languages.

**TABLE 4.12 XAP Rights Management Schema**

*The namespace prefix is xapRights. The namespace is http://ns.adobe.com/xap/1.0/rights/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| xapRights:Certificate | URL | Online rights management certificate. | External |
| xapRights:Copyright | alt Text | Legal copyright notice. The alternation is for different languages. | External |
| xapRights:Marked | Boolean | Indicates that this is a rights managed resource. | External |
| xapRights:Owner | bag ProperName | Legal owner of a resource | External |
| xapRights:UsageTerms | alt Text | Text instructions on how a resource can be legally used. | External |
| xapRights:WebStatement | URL | Location of web page describing the owner and/or rights statement for this resource. (Photoshop “Image URL”). | External |

## 4.3 Property Value Types

The following tables list the value types used in the XAP schemas. Some value types are represented by structures; where that occurs, a sub-heading of: *Name*, *Type*, and *Comments* is given to list each element of the structure.

***TABLE 4.13 Basic Value Types***

| Type | Representation | Notes |
|---|---|---|
| Boolean | *True* or *False* | The strings should be spelled as shown |
| Choice | A value chosen from a single list, and represented by a string. There may also be a vocabulary qualifier indicating the vocabulary from which the value was chosen. Note that the added vocabulary values apply to the property, not to the base vocabulary. Thus, the set of values for the property is extended; the vocabularies are not. The vocabulary qualifiers for *Choice* are:<br>● Open: Single list of preferred values, but new values may be added to the list<br>● Closed: Single fixed list of values for the vocabulary.<br>***Name*** — ***Type*** — ***Comments***<br>vQual:vocabulary — URI — Optional. The vocabulary name. The vocabulary implies a set of values, conventions, and semantics for values. The list of legal vocabulary items may be present in the schema.<br>Field namespace xmlns:vQual=http://ns.adobe.com/xap/1.0/ValueQualifier#. | |
| ColorMode | XChoice<br>(open) | A string representing the color space in a document. An open choice that includes the following:<br>RGB, CMYK, Indexed, Monotone, Duotone, Tritone, or Quadtone |
| Contact | Information about how to contact a person or organization. XAP uses the RDF Vcard (RFC 2425) representation for this structure. It includes the following, but see the IETF proposal for full details.<br>***Name*** — ***Type*** — ***Comments***<br>vCard:FN — Text — The person or organization name<br>vCard:EMAIL — alt Text — E-mail address<br>vCard:TEL — Text — Phone number<br>vCard:ADR — Text — Street, city, province, country, postal code(s)<br>Field namespace xmlns:vCard = http://imc.org/vCard/3.0# | |

| Type | Representation | Notes | |
|---|---|---|---|
| Date | YYYY-MM-DDThh:mm:ss.sTZD | A date (ISO 8601) http://www.w3.org/TR/NOTE-datetime | |
| Dimensions | ***Name*** | ***Type*** | ***Comments*** |
| | w | Real | Width |
| | h | Real | Height |
| | unit | XChoice (Open) | Examples: inch, mm, pixel, pica, point |
| | Field namespace: xmlns:stDim=http:ns.adobe.com/xap/1.0/sType/Dimensions# | | |
| Duration | ***Name*** | ***Type*** | ***Comments*** |
| | length | Real | Number of frames, or seconds |
| | unit | XChoice (Open) | Examples: *frame-count*; *elapsed-seconds* |
| | Field namespace: xmlns:stDuration = http:ns.adobe.com/xap/1.0/sType/Duration# | | |
| Font | ***Name*** | ***Type*** | ***Comments*** |
| | name | Text | Specifies the PostScript font name of the font. For TrueType fonts, the PostScript font name is obtained in the *name* table in a well-formed TrueType font. |
| | embedded | Boolean | Specifies whether the font is embedded in the document. |
| | Field namespace: xmlns:stFnt=http://ns.adobe.com/xap/1.0/sType/Font# | | |
| Integer | Signed or unsigned numeric string | Integer number representation. The string consists of an arbitrary length decimal numeric string with an option leading "+" or "–" sign. | |
| Locale | Choice (closed) | | Identifies a language.<br>Values from http://www.ietf.org/rfc/rfc1766.txt |
| MIMEType | XChoice (closed) | | image/gif, image/jpeg, etc. See:<br>http://www.isi.edu/in-notes/iana/assignments/media-types/media-types |
| ProperName | Text | | A name of a person or organization. |

| Type | Representation | Notes |
|---|---|---|
| Real | A numeric value, integer or real. Integer or decimal number of arbitrary precision. Consists of a decimal numeric string with an optional single decimal point and an optional leading “+” or “–” sign. The following qualifier can optionally appear:<br><br>***Qualifier Name*** — ***Type*** — ***Comments***<br>vQual:binRep — Text — Optional binary representation qualifier. This qualifier provides an alternate, binary representation for the number when an exact value is needed. The text is interpreted as:<br>&lt;std size&gt; , &lt;endian&gt; , &lt;hexadecimal value&gt;<br>where:<br>● *std* is the standard name. Only IEEE754 is currently supported.<br>● *size* is S for 32-bit and D for 64-bit<br>● *endian* is *L* for little-endian order, *B* for big-endian order<br>● *hexadecimal value* is the value represented in hexadecimal.<br>For example, the value might be:<br>IEEE754D,L,3A4901F387D31108<br>RDF Note: vQual:binRep is stored as a qualifier on the property node. A Real does not actually contain nested properties.<br><br>Field namespace: xmlns:vQual=http://ns.adobe.com/xap/1.0/ValueQualifier#. | |
| Resolution | ***Name*** — ***Type*** — ***Comments***<br>x — Real — Resolution in the x direction<br>y — Real — Resolution in the y direction<br>unit — XChoice (open) — dpi, etc.<br><br>Field namespace: xmlns:stRes=http:ns.adobe.com/xap/1.0/sType/Resolution | |
| Text | Unicode | A string that can contain characters from most locales. Can be marked up with additional XML tags. |
| TextEncoding | XChoice<br>(open) | Specifies the encoding of text in the document. A string from a controlled vocabulary of charset encodings. See vocabulary for *xapT:Encoding.*<br><br>**NOTE:** If multiple encodings are used in a document, the primary encoding should be specified. |
| TrackDesc | ***Name*** — ***Type*** — ***Comments***<br>start — Duration — Start time relative to doc start<br>length — Duration — Run time at normal speed | |

| Type | Representation | Notes | |
|---|---|---|---|
| *(TrackDesc, continued)* | | | |
| | bits | Integer | Bits per sample |
| | codec | Text | Name of codec |
| | description | Text | Human readable description |
| | name | Text | |
| | rate | Real | samples or fields/frames per second |
| | type | XChoice (open) | Media type (for example, audio, video, caption, etc.) |
| | blending | XChoice (open) | Name for blending algorithm |
| | Field namespace: xmlns:stTrk=http://ns.adobe.com/xap/1.0/sType/TrackDesc# | | |
| URL | URI | RFC 1630, RFC 1738, and RFC 2396 (see: http://www.w3.org/Addressing/). | |
| XChoice | A value chosen from one of several possible lists, and represented by a string. XChoice indicates that the list of choices may be extended in the schema by listing additional vocabularies. If the value is not from a core XAP schema, it's name should be specified using the vQual qualifier. Note that the added vocabulary values apply to the property, not to the base vocabulary. Thus, the set of values for the property is extended; the vocabularies are not. The vocabulary qualifiers for XChoice are:<br>● Open: Several lists of preferred values, any value can be used.<br>● Closed: Fixed list of values, but new lists can be added. | | |
| | ***Name*** | ***Type*** | ***Comments*** |
| | vQual:vocabulary | URI | Optional. The vocabulary name. The vocabulary implies a set of values, conventions, and semantics for values. The list of legal vocabulary items may be present in the schema. |
| | Field namespace xmlns:vQual=http://ns.adobe.com/xap/1.0/ValueQualifier#. | | |
| XPath | XPath | XML Path Language (XPath), for addressing parts of an XML document; see http://www.w3.org/Tr/xpath. XPath is a bag of lists; each list item must contain a single namespace and XPath element. See the example of how to set xap:Advisory values in Section 7.1.1.2, "Property Categories." | |

**TABLE 4.14 Media Management Value Types**

| **Type** | **Representation** | **Notes** | |
|---|---|---|---|
| AgentName | Text | Name of a program. Format convention: "Vendor App Version for Platform," for example: "Adobe Acrobat Distiller 5.0 for Windows" | |
| RenditionClass | XChoice (open) | The type of rendition, from a controlled vocabulary of standard names. The convention used is to have a series of ":" separated tokens and parameters, the first of which names the basic concept of the rendition. Additional tokens are optional and provide additional characteristics of the rendition. See Section 4.4, "XAP Vocabularies" for defined values. | |
| ResourceEvent | ***Name*** | ***Type*** | ***Comments*** |
| | action | XChoice (open) | Conventions to use: Use values from See Section 4.4, "XAP Vocabularies" for defined values, and to see if they apply. If new strings are used, they should be verbs in past tense. |
| | parameters | Text | Additional information |
| | softwareAgent | AgentName | Name of the software agent that was used. For example, this resource could be Edited by "Microsoft Word 7.0 for Windows," and Produced by "Adobe Acrobat Distiller 3.01 for UNIX." |
| | when | Date | Optional timestamp for when this event occurred. |
| | Field namespace: xmlns:stEvt=http://ns.adobe.com/xap/1.0/sType/ResourceEvent# | | |
| ResourceRef | An identifier for a resource; used to uniquely identify a resource. | | |
| | ***Name*** | ***Type*** | ***Comments*** |
| | documentID | URI | See comments in xapMM Media management schema |
| | versionID | Text | |
| | RenditionClass | Text | |
| | Field namespace: xmlns:stRef=http://ns.adobe.com/xap/1.0/sType/ResourceRef# | | |

| **Type** | **Representation** | **Notes** |
|---|---|---|
| URI | Text | The purpose of this URI is to provide a unique identifier in the widest possible scope (like all time and space). Any valid URI can be used.<br>For DocumentID URIs we recommend the following convention:<br>vendor:docid:{manager}:{uri_safe_text}<br>where:<br>*vendor* is replaced by a short representation of the company name implementing the tool that is assigning the name. For example, “adobe.”<br>*{manager}* is replaced by a short and unique application identifier (most Adobe applications have three-letter codes through Adobe Online, or you can use Apple Creator codes), and {uri_safe_text} is replaced by additional id information, as long as it is encoded in URI safe text. For example, it may contain a 128-bit UUID encoded as hex.<br>Example (Adobe Acrobat):<br>adobe:docid:acro:9705C6F1-81BD-11d3-9CCC-006097CEC6D4<br><br>(See http://www.w3.org/Addressing/ for information on URIs) |

| **Type** | ***Name*** | ***Type*** | ***Comments*** |
|---|---|---|---|
| Version | comments | Text | Comments concerning what was changed |
| | event | ResourceEvent | High level, formal description of what operation the user performed |
| | modifyDate | Date | Date this version checked in |
| | modifier | ProperName | Person who modified this version |
| | version | Text | Version name, such as “1.2” |
| | Field namespace: xmlns:stVer=http://ns.adobe.com/xap/1.0/sType/Version# | | |

***TABLE 4.15 Adobe Support Metadata Value Types***

| Type | Representation | Notes | |
|---|---|---|---|
| FileDisposition | ***Name*** | ***Type*** | ***Comments*** |
| | filename | Text | Simple local filename in local os form |
| | OS | Text | Examples: UNIX, MacOS, Windows |
| | directoryPath | Text | Fully qualified pathname to directory |
| | Field namespace: xmlns:stDsp=http://ns.adobe.com/xap/1.0/sType/FileDisposition# | | |

***TABLE 4.16 Basic Job/Workflow Value Types***

| Type | Representation | | |
|---|---|---|---|
| Job | ***Name*** | ***Type*** | ***Comments*** |
| | name | Text | Informal name of job. This name is for user display and informal systems. |
| | id | Text | Unique Id for the job. This field is a reference into some external job management system. |
| | Field namespace xmlns:stJob=http://ns.adobe.com/xap/1.0/sType/Job# | | |

## 4.4 XAP Vocabularies

Vocabularies provide a means of specifying a limited but extensible set of values for a property. The metadata schema can indicate whether the set of legal values is fixed or can be extended. The flexibility can be limited in the following ways:

- Fixed list of values (closed vocabulary, choice type)
- Fixed list of values, but new lists can be added (closed vocabulary, xchoice type)
- List of preferred values, but any value can be used (open vocabulary, choice type)
- Several lists of preferred values, but any value can be used (open vocabulary, xchoice type)

If a property value is to have a very definite meaning and all users of that property must know the exact meaning, the first, most restrictive form should be used. If there are well defined sets of values whose meanings are known, but additional value might be used without causing problems, then the less restrictive forms may should be used.

Unless otherwise specified, the default namespace for all vocabulary elements is:

http://ns.adobe.com/xap/1.0/corevocabulary

Table 4.17 lists the XAP vocabulary elements.

***TABLE 4.17 XAP Vocabularies***

| **Vocabulary for** | **Vocabulary elements** | |
|---|---|---|
| xapDyn:Tracks/*/stTrk:blending | ***Value*** | ***Comment*** |
| | copy | |
| | alpha_blend | |
| | transparent | |
| | dither | |
| xapG:ColorSpace | ***Value*** | ***Comment*** |
| | RGB | |
| | CMYK | |
| | indexed | |
| | monotone | |
| | duotone | |
| | tritone | |
| | quadtone | |
| xapG:Compression<br>xapDynA:Compression | ***Value*** | ***Comment*** |
| | LZW | |
| | JPEG | |
| | Huffman | |

| Vocabulary for | Vocabulary elements | |
|---|---|---|
| xapG:GraphicsType | ***Value*** | ***Comment*** |
| | raster | |
| | vector | |
| | dynamic | |
| xapT:Encoding | Specifies the encoding of text in the document. | |
| | ***Value*** | ***Comment*** |
| | ASCII | |
| | ISO-8859-n | Specifies the ISO encoding for Latin fonts, according to ISO 8859; *n* specifies the part of ISO 8859. (ISO 8859 has 12 parts; for example, ISO 8859-1 is one of several encodings for Latin character fonts, and ISO 8859-5 is for Cyrillic fonts.) |
| | Mac | Encoding for Macintosh Roman character set. |
| | WinAnsi | Encoding for Windows Roman character set. |
| | UTF-8 | Unicode one-byte |
| | UTF-16 | Unicode two-byte encoding. Document text includes a BOM (Byte-Order Marker) to indicate byte order. |
| | UTF-16BE | Unicode two-byte encoding; big-endian byte order; (BOM not in document text). |
| | UTF-16LE | Unicode two-byte encoding; little-endian byte order; (BOM not in document text). |
| | UCS-2 | see ISO10646 |
| | UCS-4 | see ISO10646 |
| | EUC-CN | Simplified Chinese |
| | GBK | Simplified Chinese |
| | Big_Five | Traditional Chinese |
| | EUC-TW | Traditional Chinese |
| | Shift-JIS | Japanese |
| | EUC-JP | Japanese |

| Vocabulary for | Vocabulary elements | |
|---|---|---|
| *(xapT:Encoding, continued)* | EUC-KR | Korean |
| | Unified_Hangul_Code | Korean |
| | custom | Font has a custom encoding that does not conform to standard encodings. |
| xap:Locale<br>dc:language | Values used in this vocabulary are from Locale, http://www.ietf.org/rfc/rfc1766.txt. | |
| xap:Format<br>dc:Format | Values used in this vocabulary are those from MIME Type. See the relevant standard for the current values. http://www.isi.edu/in-notes/iana/assignments/media-types/media-types | |
| xapG:NaturalDimensions/stDim:unit<br>xapGImg:Dimensions/stDim:unit<br>xapDynV:Dimensions /stDim:unit<br>xapTPg:MaxPageSize /stDim:unit<br>xapGImg:Resolution | ***Value*** | ***Comment*** |
| | inch | |
| | mm | |
| | pixel | |
| | pica | |
| | point | |
| xapDyn:Tracks.type | ***Value*** | ***Comment*** |
| | audio | |
| | video | |
| | image | |
| | caption | |
| | sprite | |
| | text | |
| | href | |

| Vocabulary for | Vocabulary elements | |
|---|---|---|
| xapMM:RenditionClass | ***Value*** | ***Comment*** |
| | default | Indicates the master document. No additional tokens allowed. |
| | thumbnail | For a simplified and/or reduced preview of a version. Additional tokens, if any, provide more characteristics of the thumbnail. The colon character “:” is used as a delimiter. The recommended order is: thumbnail:format:size:colorspace.<br>Examples:<br>thumbnail:jpeg<br>thumbnail:16x16<br>thumbnail:gif:8x8:bw |
| | screen | For a screen resolution/Web rendition |
| | proof | For a review proof |
| | draft | For a review rendition |
| | low-res | For a low resolution, full size stand-in |
| xapMM:History/*/stEvt:action | ***Value*** | ***Comment*** |
| | converted | Format changed in some way. |
| | copied | |
| | created | |
| | cropped | |
| | edited | Covers all kinds of editing, generally of the content. |
| | filtered | Content filtering algorithm applied. |
| | formatted | |
| | version_updated | |
| | printed | |
| | published | |
| | managed | Placed under media management. |
| | produced | |
| | resized | |

# 5 Metadata Property Commentary

## 5.1 Metadata Properties

Table 5.1 through Table 5.4 list the properties of the Adobe XAP metadata schemas, and explain the use of, and rationale for, each property.

***TABLE 5.1 XAP Core Schema Metadata Properties***

| Property | Description, Notes, Rationale |
|---|---|
| xap:Advisory | If the value of any external or relational properties were changed outside the authoring application, the value of xap:Advisory is a *bag* of lists that point to those properties. Each list should contain a single namespace and XPath. See the example of how to set xap:Advisory values in Section 7.1.1.3, "XAP:Advisory Example." |
| xap:Author | No structure should be assumed, but see Dublin core definition: "Family, Given."<br>People for whom the credit of authoring this resource is attributed.<br>The Author values are strings representing an author. Although recommended conventions exist for the structure of the name, programs should not make any assumptions about the structure or format of the ProperName.<br>Searches should do unanchored matches against the name strings. The strings can include XML/HTML formatting and other information.<br>Authoring applications should set the value of the xap:Author property as directed by the user or other conventions used to establish authorship. The user should be allowed to edit this resource.<br>Aggregating applications: Insertion or deletion of a document should not affect author information for either the inserted or containing document.<br>If you want to have multiple authors, use dc:creator. |
| xap:BaseURL | If this document contains Internet links, and those links are relative, they are relative to this base URL.<br>The purpose of this property is to provide a standard way for embedded relative URLs to be interpreted by tools. Web authoring tools will want to set the value based on their notion of where URLs will be interpreted.<br>Should be the same as the value of the path property of the FileDisposition if the value of the OS attribute tag is "URL." |

| Property | Description, Notes, Rationale |
|---|---|
| xap:CreateDate | The date and time the document was originally created.<br>One meaning of the date is when the first *Save* or *New* was executed for a new document. Another meaning is the date that the content was created (for example, an illustration that has been scanned). The user would need to set the CreateDate explicitly in that case.<br>Once set, tools should not change the value, unless explicitly directed to do so by the user. |
| xap:CreatorTool | Original tool that was used to create the resource (at least the first known tool).<br>If history is present in the metadata, this value should be equivalent to that of xapMM:History's softwareAgent property. |
| xap:Description | An account of the content of the resource.<br>A textual description of the resource content. The alternation is for different languages.<br>Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.<br>The description text is entirely under user control. Tools can set the default to some summary if they wish but should not overwrite user-entered information. The XML strings may contain HTML formatting tags and other information.<br>The xml:lang attribute on the property alternative, if present, is used to identify the language on each alternative. |
| xap:Format | Defines the resource format (e.g. video, audio, etc.)<br>Tools and Application should set this resource to the save format of the data.<br>Recommended best practice is to select a value from a controlled vocabulary (specifically, the list of Internet Media Types [MIME] defining computer media formats). |
| xap:Keywords | Can consist of an unordered list of descriptive phrases, or specify the topic of the content of the resource.<br>Typically, a Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource.<br>Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.<br>This resource is user editable. It is unlikely that the applications would change the value of Keywords without explicit user direction.<br>Keywords are not merged when another document is embedded by an aggregating application. |

| Property | Description, Notes, Rationale |
|---|---|
| xap:Locale | Default locale for otherwise-undefined-locale elements of the resource. Applies to the entire resource.<br>This property should be set and managed by applications. When user actions or system changes indicate a new locale, the next save or write should update this property.<br>1. Recommended best practice for the values of the Language element is defined by RFC 1766 which includes a two-letter Language Code (taken from the ISO 639 standard), followed optionally, by a two-letter Country Code (taken from the ISO 3166 standard).<br>The following are examples of language codes: ‘en’ for English, ‘fr’ for French, or ‘en-uk’ for English used in the United Kingdom. |
| xap:MetadataDate | Date and time that any metadata for this resource was last changed.<br>This property may be used to optimize searches by excluding resources that might be out-of-date with respect to externally stored metadata. When a reconciliation is executed between an external metadata store and the XAP metadata, the modify time on the XAP metadata can be compared with the modify time bounds on the external metadata, and, if the external metadata is clearly newer, its value can be merged into the XAP metadata. If the external metadata has been updated but it cannot be established that it is newer then the MetadataDate, the property-by-property timestamp comparisons may be necessary. |
| xap:ModifyDate | The date and time the resource was last modified.<br>Media management systems receiving resources via check-in that have no value for ModifyDate may set the value to the check-in timestamp.<br>It is not necessary for applications to update this property each time an opened resource is modified (that is, for individual edits). An acceptable interpretation of this property would be to update it at the last resource *write* (or *save*) time. |
| xap:Nickname | Short informal name for resource, e.g. “Adobe logo,” or “Feb issue,” – for UI purposes.<br>A user specified value should not be overwritten by an application wishing to set the value. The value can be used for window/pane naming or other similar purposes.<br>Applications may set the value if there is some convention for this type of information in place and no user-supplied value is present. |

| Property | Description, Notes, Rationale |
|---|---|
| xap:Rights | Information about rights held in and over the resource.<br><br>**NOTE:** This property is deprecated; the use of xapRights:Copyright is recommended for use in its place.<br><br>Inline text of copyright or other rights statements. The alternation is for different languages. See the xapRights schema for structured rights statements.<br><br>Typically, a Rights element will contain a rights management statement for the resource, or reference a service providing such information. Rights information often includes intellectual property rights (IPR), copyright, and various property rights.<br><br>If the Rights element is absent, no assumptions can be made about the status of these and other rights with respect to the resource.<br><br>Authoring applications: this field is primarily for user use. User-entered data should not be overwritten. The value is not intended for automatic processing and programs should not make assumptions about the format of the data.<br><br>Aggregating applications: when documents are aggregated, the rights of the aggregate document could be affected, but often the meta data is not merged, so it remains separate. A tool must be used to extract the rights of all of the documents in an aggregate and produce a merged value for the user. |
| xap:Title | A name given to the resource. The alternation is for different languages.<br><br>Typically, a Title will be a name by which the resource is formally known.<br><br>The Title values are arbitrarily formatted strings. Programs should not make any assumptions about the structure or format of the Title value. The strings can include XML/HTML formatting and other information. The language can be determined by looking at an xml:lang property in the string, if present. For example:<br><br>`<rdf:li xml:lang="de">`<br>`  Ausdehnbares Schreiben und Veröffentlichen Metadata (XAP)`<br>`</rdf:li>`<br><br>Authoring applications: should set the value of the xap:Title resource as directed by the user or other conventions used to establish the title. The user should be allowed to edit this resource. |

**TABLE 5.2 Adobe Media Management Metadata Properties**

| Property | Description |
|---|---|
| xapMM:ContainedResources | References to resources that may be contained in this (aggregate) document version.<br>The resources are likely contained in this document but may have been deleted. Thus, this list is a likely to be a superset. The intent is to make it easy to locate references to documents that are embedded in other documents without placing a large management burden on applications and tools.<br>Since ResourceRefs are provided, it should be possible to locate the inserted documents if they exist by using a media management system or module. |
| xapMM:ContributorResources | Resource that in some way contributed to this one, implying a dependency.<br>The resources listed in ContributorResources are intended to be dependencies for the listing resource. Although the resources are not embedded in their entirety, some component of them may be embedded or there may be some dependence such that changing one of the resources listed in ContributorResources would imply a possible change to this resource.<br>Since ResourceRefs are provided, it should be possible to locate the referenced documents if they exist by using a media management system or module.<br>Tools or users would need to have some way of indicating such a dependency. Applications may be able to set this resource depending on specific cases of user actions. |
| xapMM:DocumentID | The common identifier for all versions and renditions of a document. This creates a family of resources.<br>This identifier is a string representation of unique information associated with the document (such as creation timestamp or random number). There is a specific syntax for DocumentID, see URI on page 41.<br>The DocumentID should be assigned when a new document is created, typically when File:New (or equivalent) is executed. Subsequent edits should not affect this value.<br>Media management systems and tools must not change the DocumentID. The only exception is that media management systems, tools, and aggregating applications may assign a document ID if there is none.<br>Do not copy the xapMM:DocumentID value of an imported document into current document. |

| Property | Description |
|---|---|
| xapMM:History | A chronological sequence of high-level user actions which resulted in this resource.<br><br>This is intended to give human readers a general indication of the steps taken to make the changes from the previous version to this one. The list should be at an abstract level. It is *not* intended to be an exhaustive, keystroke or other detailed history.<br><br>Example:<br><br>Action: Created Parameters: SoftwareAgent: Adobe Photoshop 5.5 for Windows When: …<br>Action: Cropped Parameters: old size 640x480 SoftwareAgent: Adobe Photoshop 5.5 for Windows When: …<br>Action: Resized Parameters: SoftwareAgent: Adobe Photoshop 5.5 for Windows When: …<br>Action: Edited Parameters: SoftwareAgent: Adobe Photoshop 5.5 for Windows When: … |
| xapMM:LastURL | Last place this resource was written.<br><br>Each time the resource is written to a file or web server, the name of the file or server location should be stored in this property. The URL syntax handles both file and web locations.<br><br>The purpose of this property is to enable some amount of tracing of the history of where the document has been. Copies of the document retain the LastURL value from when the tool wrote it. If the file is moved, there is some means to trace back to where it was last written. |
| xapMM:Manager | Name of the software agent that manages this resource.<br><br>Although AgentName has a defined format, this property is intended for human user use. It tells the user where to go look to find things, but does not really enable automated retrieval or check-in of documents.<br><br>This property is set by media management tools and may be set at check-out time or earlier. |
| xapMM:ManageTo | Specifies the location of managed rendition of asset. |
| xapMM:RenditionClass | Rendition class name for this resource. This property should be absent or equal to “default” for a document version that is not a derived rendition.<br><br>Each version of a document may have several renditions. Each rendition is distinguished by a rendition class name. This property lists the rendition class name for a particular rendition.<br><br>If the application UI is capable of determining that a rendition is being created, then the type of rendition should be set by setting a value for xapMM:RenditionClass in the new file. |

| Property | Description |
|---|---|
| xapMM:RenditionOf | For a rendition of something, indicates the resource that this resource is a rendition of.<br>This property indicates what resource is the source of this rendition.<br>When a rendition is created, the creating application should set RenditionOf to indicate the resource from which this rendition is derived. The ResourceRef type allows any resource to be specified as the source for this rendition, although the DocumentID and VersionID would normally be the same in the source and the rendition. |
| xapMM:SaveID | An id number which is incremented each time the resource is written to a particular URL.<br>This number is used to help track the movement of a resource through media management and file systems. Each time an application (or other tool) writes the resource out to the same location, this property value should be incremented.<br>If it is being written to a different location than the LastURL property indicates, the SaveID should be reset to 1 and the LastURL property updated to reflect the new location. A History entry should also be added indicating this. (See XapMM:History) |
| xapMM:VersionID | The document version identifier for this resource.<br>Each version of a document gets a new identifier. Usually these values are simply incrementing integers 1, 2, 3 . . . etc. Media management systems may have other conventions or support branching which requires a more complex scheme. The Version identifier should be kept short.<br>This property should be used primarily by a media management system. Applications with sufficient interfaces to detect user intent of creating new versions (for example, Microsoft Word®), can assign new version identifiers at appropriate times, but should be careful to avoid conflicts with media management systems. |
| xapMM:Versions | The version history associated with this resource. Entry [1] is the oldest known version for this document, entry [LAST] is the current version.<br>Typically, a media management system would fill in the version information in the metadata on check-in. Individual applications can also place version information into metadata, but need to be careful to avoid conflicts with media management systems. An application should avoid adding version information if a media management system is in use.<br>It is not guaranteed that complete history of version from the first to this one will be present in the xapMM:Versions property. Interior version information may be compressed or eliminated and the version history may be truncated at some point. |

***TABLE 5.3 Adobe Support Schema Metadata Properties***

| Property | Description |
|---|---|
| xapS:EntityTag | Supports cache validation when comparing the *content value* of two blocks of metadata. Follows a convention based on the *entity tag* concept defined in HTTP/1.1. See http://www.ietf.org/rfc/rfc2616.txt |
| xapS:FileDisposition | Preferred file name to save on disk, or preferred URL for publishing on a web server, by OS.<br>If absent, this resource can be set to the *Save As* filename. If already set, it should be left alone. The user can explicitly edit it if desired. It can be used, at a tool's discretion, as a default for *Save As*, *Export*, or check-in operations.<br>Attribute tags can be used to specify different values for different systems and environments. The following conventions should be used: *Windows*, *MacOS*, *Unix*, or *URL*<br>New tags can be used if the above are not adequate. |
| xapS:ResourceID | The unique identifier for this particular resource. Used for storage systems which provide a unique identifier for a stored resource which is incompatible with the format defined in xapMM.<br>This property is for use by media management systems. When used, the values should meet the following requirements:<br>This identifier needs to be unique in time and space. It can be a random number or can include some information uniquely associated with the document (such as creation timestamp).<br>ResourceID should be set when a new resource is being created. This would typically be done by a media management system on check-in.<br>In absence of a media management system this property should not be used. Authoring application should not set or use it.<br>A media management system is allowed to change the ResourceID value for a document when the document is checked-in. Thus, a media management system may implement its own notion of unique identifiers for resources and override the value that was placed by a different media management system. |
| xapS:Size | File size in bytes.<br>This value should be the actual file size and include any embedded metadata.<br>This property would typically only be set by media management systems and used by clients of media management systems. It should be deleted by applications when a file is open to avoid confusion of the "current size" versus the last saved size. |

**TABLE 5.4 *XAP Basic Job Properties***

| Property | Description |
|---|---|
| xapBJ:JobRef | Name of the job(s) that this document is part of. Use of job names is under user control. Typical use would be to identify all documents that are part of a particular job or contract.<br><br>There are multiple values since there may be more than one job using a particular document at any time and it may be desired to keep historical information about what jobs a document was part of. |

# 6 XAP Extensibility

## 6.1 Making Custom Schema

The schemas defined by this document are *core schemas* that all implementations of XAP must support. If your metadata needs are not already covered by the core schemas, you may add your own schemas as extensions.

The XAP metadata framework was designed to be easily extensible, particularly for the addition of custom schemas. All software systems that support XAP must handle extension metadata.  A conforming software system must be able to parse extension metadata, read/modify/write extension metadata, and serialize extension metadata back into XAP format.

To define a new schema, you must write a human-readable schema specification document Make this specification document available to any developers who need to write code that understands your metadata.

NOTE: Future versions of XAP may include support for machine-readable schema specifications, but such support will always be in addition to the requirement for human readable schema specification documents.

Your specification document must include at least two items: 1) a unique name for your schema, in the form of a URI, and 2) a table which represents the name of each property, the value type, the description of the property, and its usage type.

NOTE: The XAP 1.0 implementation does not support aliasing for custom schemas. Usage of aliasing for custom schemas has the potential to result in data loss or software failure.

If you define properties that have structured value types, you may also need additional URI names to identify the components of a structured property value (for a core schema example, see xapG:NaturalDimensions in Table 4.2, "XAP Graphics Schema" on page 29).  This is only necessary if the value type is used in more than one property definition.  If not, the URI you choose for the schema will be used for all component property values as well.

For example, if you are working on the JAKES project for Billingsgate.net, you might select for your schema name the URI *http://ns.billingsgate.net/JAKES/*. This does not have to be an actual URL that people can connect to on the Internet.

Once you have selected a unique name, you should also pick a short prefix. This will be used to qualify your property names (see XML namespace specification). Remember that XML implementations are free to replace your prefix name with another, but when there is no other choice, you should have a default name selected. For example, again referring to the JAKES project, you might pick "jks:" as the prefix.

Now define all desired properties, using this reference document as a guideline for the types of structures and values you can define. For example, if JAKES needs a Quay property, which has a simple text value, you would define:

**Table X.X JAKES Schema** (example of schema extension)

*The namespace prefix is jks. The namespace is http://ns.billingsgate.net/JAKES/.*

| Property | Value Type | Description | Category |
|---|---|---|---|
| jks:Quay | Text | Place to dock | External |

You can then add more properties as needed, following the RDF and XAP syntax requirements described in this document to create compatible RDF metadata.

## 6.2 New Versions of Existing Schemas

We have already seen how versions of schemas are introduced; they are part of the URL which defines a name space. The following convention for schema versions is recommended for conforming applications.

Convention: If and only if a client application implements schema versions as fully backward compatible, we recommend that Properties be set *once* in the group associated with the earliest schema version in which they are defined. Full backwards compatibility means that schema N+1 contains all of schema N, with no changes other than new *additional* Properties. This case is exemplified by the example above: our hypothetical Dublin Core 9.0 contains all of the previous version of the Dublin Core schema, without change, except that we add the new element CRYPTOKEY. A conforming application may also define a new Property which is intended to supersede an old one. For example, suppose schema N defines a PageNumber property, whose value is a number. Later, we discover that page numbers can be arbitrary strings, like “xix” or “A-3”. For schema N+1, we define a new property ExtendedPageNumber, whose value can be a String. We continue to set the old property when appropriate, but when the page number needs to be a string, we omit the old one and only set the ExtendedPageNumber.

To add elements in a later version of a schema, just do the same thing as you would for adding elements in a different schema. Consider the hypothetical Dublin Core version 9.0, which contains the element CRYPTOKEY:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description about="" xmlns:dc="http://purl.org/dc/elements/1.1/"
        dc:creator="John Doe"
        dc:title="Adobe XAP Metadata Framework"/>

    <rdf:Description about=""
```

```
        xmlns:xap="http://ns.adobe.com/xap/1.0/"
        xap:Format="text/html" xap:Locale="en"/>

      <rdf:Description about=""
                   xmlns:dc="http://www.purl.org/dc/elements/9.0/"
         dc:CRYPTOKEY="29J39LJKS9JL388X,39872KD987KJ3O987SO8NV83LU784"/>
  </rdf:RDF>
```

Note that we are allowed to reuse the *dc:* prefix if we want, since we are in a different rdf:Description, and thus a different xmlns scope.

# 7 Application Integration Guidelines

## 7.1 Supporting XAP Metadata

This section describes what applications need to implement support for XAP Metadata. It explains what information should be considered metadata; what is needed for a User Interface; and how to store data. It also describes what metadata actions need to be performed for specific application operations like *Open*, *Save As*, and embedding an image in a document.

### 7.1.1 Requirements

To properly support XAP metadata, the following three rules must be followed.

#### Preserve Metadata

Any metadata that is read as part of loading application data must be preserved and written back out when the application data is rewritten. The metadata must be preserved regardless of whether the application understands its semantics.

When application data is loaded from one format and written out as a different format, metadata should be preserved by including it in the file in the saved format, or writing it to a separate, parallel file if the new format does not accommodate metadata.

A standard library is being developed to store metadata in the appropriate place in a variety of formats.

#### Set Meaningful and Useful Information in Metadata

Each application will have a set of metadata properties that are relevant to that application and useful to customers who use metadata. The application should set metadata values for those properties. This would be done at least at save-time and, if the application provides an interface to access or edit metadata, at points where those interfaces could be used. This might mean that the metadata would need to be updated each time the underlying application data, to which the metadata is related, is changed.

One important rule is that you should not define a property in the metadata if you can’t find a value for it that makes sense. An example of this would be the properties for the XAP Rights Management Schema; if you don’t have valid values for those properties, those properties should not be included.

There is a fine line between metadata and application data. Ultimately, it is a judgment call as to what to include in metadata. Here are some useful guidelines for making this judgment:

- Information that is only useful to the internals of the application is not a good candidate for public metadata.
- Information that can only be used by a tool that is deeply involved with application internals is not a good candidate for metadata.
- Information whose size or structure has a linear relationship with the size of the application data is not a good candidate for metadata. It will be too large and impractical to maintain.
- Information that is only useful in conjunction with other application data that is not available in metadata is not a good candidate for metadata.
- Information that users or tools may want to browse, select, or otherwise use to characterize an asset is a good candidate.
- Explicit user information that the application currently supports is a good candidate for metadata.
- Information that the application happens to have that is present in some industry metadata standard is a good candidate.

#### Allow display/edit

If the application has metadata or it is a user expectation to be able to look at or edit or enter metadata, then the application should provide a metadata display and editing interface.

A key requirement for such an interface is that it accommodate the extensibility model of XAP, namely that metadata defined and introduced after the product ships receive the same status as metadata defined by Adobe or before the product has shipped. Metadata properties defined later may be as or more important to the customer than metadata defined prior to shipment.

Fully general display and editing of metadata is complicated by the existence of sequence and structured values, and, particularly, sequences of structured values. The user interface may have to support more than simple (name, value) pairs.

### 7.1.1.1 Aliasing

When aliased properties are serialized, the linked value should be copied. When aliased properties are parsed, their properties should be compared to make sure they are the same (byte-for-byte text comparison). If they are not, a conflict error should be raised.

Aliasing of schema extensions cannot be supported in the absence of an automated schema management system, and hence is not supported in XAP Metadata 1.0. Any schemas not listed in this document are not supported.

### 7.1.1.2 Property Categories

XAP metadata may be read by an application when it opens a resource. For the results of modifying metadata associated with a resource to give predictable behavior, it is important that creator applications follow some simple guidelines. These are usage guidelines and are not enforced by software; in situations where they do not make sense, they may be ignored. The interpretation of the XAP properties is broken down into the following categories:

*Internal*— reflection of the information contained within the resource. Modifications to internal properties are ignored by the authoring application.

*External* — Metadata consisting largely of annotation information. Modifications may be displayed by the authoring application but are not acted upon.

*Relational* — This is a subset of internal metadata that pertains to the relationship of this resource with outside systems such as management markers or print resolution. Modifications to these properties outside the authoring application should be "resolved" against the internal information in the document.

#### 7.1.1.3 XAP:Advisory Example

If External or Relational properties are modified outside of the authoring application,they should be referenced by the xap:Advisory property. The value of that property is a *bag* of lists, and each list element contains a single namespace and XPath element.

For example, let's assume that a resource has had its dc:format, xapG:NumberOfColors, and xapGImg:Resolution/stRes:unit properties modified externally, so those properties need to be marked as *Advisory*: The following example illustrates how that would be done in RDF:

```
    <rdf:Bag>
      <rdf:li>
          http://purl.org/dc/elements/1.1/ format
      </rdf:li>
      <rdf:li>
          http://ns.adobe.com/xap/1.0/xap/g/ NumberOfColors
      </rdf:li>
      <rdf:li>
          http://ns.adobe.com/xap/1.0/xap/g/img/ Resolution/stRes:units
      </rdf:li>
    </rdf:Bag>
  </xapMM:Advisory>
```

Notice that the two parts, the namespace and the property's XPath, must be separated by at least one character of whitespace.

## 7.2 Storage of Metadata

The recommended place to store metadata is embedded within the application file itself. This ensures that the metadata and application data will not become separated. There is basically one exception to this: when files are being optimized for minimum-size delivery on the internet. In that case metadata should be minimized or eliminated altogether. The XAP metadata is stored in a XAP Packet, as described in 3.8, "XML Packets." Because it is possible to have non-XAP data in a XAP/XML packet, care must be taken to not overwrite that data.

## 7.3 Metadata Actions For Specific Application Operations

The following sections describe which metadata actions should be performed for specific application operations such as *Open*, *Save* and *Save-As*, creating new versions and renditions, placing images, and embedding unaltered copy.

### 7.3.1 General Comment on Schemas

Applications should not assume that the only properties that they will encounter in a XAP metadata object are those defined in the current schema or that all properties defined in the schema are present.

If schema information for a specific property is not available (not in the schema, or there is no schema), then primitive values are considered simple text. Structures and containers such as sequences and bags still function without any problems because their representation is self-describing.

#### 7.3.1.1 New Document

When a new document is created, the following metadata properties should be set by the creating application:

- xap:CreateDate
- xap:Locale
- xapMM:DocumentID
- xapMM:VersionID (value = 1)
- xapMM:RenditionClass (value set to “default”)
- xapMM:History (add first entry)

#### 7.3.1.2 Save and Save-As

When a new document is first saved, the following additional metadata properties should be set by the creating application:

- xap:Author (to default, if available and not already set)
- xap:ModifyDate
- xap:MetadataDate
- xap:Format
- xap:Title (to default, if available and not already set)
- media-specific metadata properties as applicable
- xapMM:LastURL
- xapMM:SaveID
- xapMM:Manager
- xapBJ:JobName

- xap:Locale

If there are other pieces of metadata redundant with XAP in the application data (typically for legacy reasons), it must be made consistent with XAP at save time.

Individual applications should set the following metadata properties:

- xap:CreateDate
- xap:Format
- xap:Locale
- any media-specific properties
- general metadata such as xap:Author, xap:Description, and xap:Title
- xapMM:ContainedResources to track embedded documents
- xapMM:History (for major impact changes in the document)

#### 7.3.1.3 Versions

Creation of a new version usually requires some explicit action on the part of the user and consequently there must be some user interface that allows the application to deduce that a new version is to be created. This would commonly be the check-in operation for an authoring application or a media management system.

From the point of view of XAP metadata, all that is required is to assign a new xapMM:VersionID typically by incrementing it. The application creating the new version is responsible for assigning the new VersionID.

#### 7.3.1.4 Renditions

Renditions of a version of a document are created directly by some explicit user action or indirectly by some tool that is carrying out some user action. Renditions may be stored as normal resource or files, or may be embedded inside another resource by an aggregating application.

Creation of a rendition involves changing the xapMM:RenditionClass to a value that identifies the kind of rendition being created, and changing the c property to indicate the resource from which the new rendition is derived.

If an aggregating application creates a new rendition of a resource as part of embedding a resource, the metadata for the modified resource should be changed as indicated in the previous paragraph and the xapMM:ContainedResources property of the aggregate resource should be modified to include the new, embedded rendition.

#### 7.3.1.5 Document *Open* Time

When documents are opened, there is some processing that is required to reconcile metadata values that may have been edited in various locations. At *open* time, applications need to analyze and merge metadata as detailed below.

Metadata sources include the following:

- XAP metadata stored with the application document representation
- XAP metadata stored in a separate metadata repository
- non-XAP metadata stored with the application document representation
- non-XAP metadata stored in a separate metadata repository

At *open* time, any internal metadata that has been edited outside the application should be reset to values consistent with the application data. This is effectively recovering from the editing of what should have been read-only metadata.

Metadata conflicts should be resolved using timestamps to pick the latest updates which may have been made from different sources.

Thus, the following steps should occur when the document is opened:

1. XAP metadata is extracted from the file and processed.
2. Metadata is obtained from the media management system, if available.
3. Media management system metadata is merged into the application file metadata using property timestamps to identify metadata items that have been modified, and if necessary, choosing the latest value according to an application-defined algorithm when conflicting changes have been made.

When and if the file is re-saved by the user, the merged metadata is written out.

## 7.3.2 Media Management System Actions

The media management system should set the following metadata properties:

- xapMM:Manager
- xapMM:Versions (the media management system may truncate the length of the version history that is embedded in the document metadata to keep it from growing without bounds)
- xapMM:VersionID
- xapS(all) (as needed)
- xapMM:ManageTo

Media management systems likely have their own database for storing metadata about managed documents. It is the responsibility of the media management system to extract XAP metadata and merge it with its internal database to maintain data integrity. Similarly, it is the media management system’s responsibility to supply updated XAP metadata to applications so that embedded metadata can be properly maintained and reflect changes made to metadata by other applications and tools while the document is not being actively edited.

### 7.3.3 Document Embedding and Metadata Preservation

This section contains guidelines on how applications should handle the embedding of one document, or image, in another document.

**NOTE:** The guidelines given here about how to handle different kinds of embedding are based on a model that reflects the conceptual document model on which XAP is based. Implementors who follow these guidelines must consider how the specific operational embedding model, inherent in their application, aligns with the guidelines described here to determine which embedding procedures to utilize. In general, the preference should always be to preserve as much of the original metadata as practical; implementors are encouraged to consider whether they can enhance their implementation's embedding model so as to make this possible.

#### 7.3.3.1 Placed-Image Metadata

A placed image refers to the embedding of all or a part of one document (the *contributor document*) into another document (the *aggregate document*). The term *placed image* is used because the contributor document often contributes an image, but the concept can apply to any content that is contributed from one document and embedded in another. The application that does this is called the aggregating application. Examples of placed images include placement of a page or image into a PDF file by Acrobat®, or placement of an article, page, or image on a page by Adobe InDesign™ or Framemaker®.

When a placement of content from a contributor document into an aggregate document occurs, one of four things should happen with metadata, according to the guidelines in the following sections. This does not apply when a reference to another document is placed into an aggregate document. When a reference is added:

- The contributor document metadata remains with it.
- The xapMM:ContributorResources property in the aggregate document should have a ResourceRef to the contributor document appended to it. If there is xapMM:DocumentID metadata present in the contributor document, it should be used to form the ResourceRef. If not, a ResourceRef should be constructed using a URL to the file containing the contributor document.

#### 7.3.3.2 Full Unaltered Copy Embedding

When the contributor document is embedded in the aggregate document unaltered and in its entirety, all metadata should be copied with the contributor document. Examples of this form would include placement of an image from a contributor document that contains only that image into an aggregate document without alteration. A more subtle example would be the placement of an image that is part of an aggregate document but was at some point in the past a complete document into another aggregate document.

The underlying principle at work in this case is that the content and metadata of the placed document are not being altered; only the storage location changes and this does not affect either content or metadata.

Typically, an application would not need to take any action to implement this case because the contributor file contents that are embedded in the aggregate document already include the metadata.

In addition, in the document level metadata for the aggregate document, xapMM:ContainedResources should include a reference to each embedded resource. Specifically, the xapMM:ContainedResources property in the aggregate document should have a ResourceRef to the contributor document appended to it. If there is xapMM:DocumentID metadata present in the contributor document, it should be used to form the ResourceRef. If not, a ResourceRef should be constructed using a URL to the file containing the contributor document.

#### 7.3.3.3 Subset or New Rendition Embedding

When a subset of the content of the contributor document is embedded structurally in an aggregate document or when the form or rendition of the content from the contributor document is changed from the original, some metadata should be included with the placed content. Examples of this would include placement of a page from a multi-page contributor document into an aggregate document or the alteration of the form of an image (such as the size, colorspace, or resolution) as it is embedded into an aggregate document.

When a component (or possibly all) of one document is embedded in another document, the following metadata properties should be set. This list represents a minimum. Ideally, all of the metadata should be preserved, but this will not always be practical because of size and granularity constraints.

When a component of a document is placed into another document, the component maintains its own metadata. The aggregate document also maintains its own metadata. The philosophy is that there should be at least minimal identifying information on the placed component so that its origin and characteristics could be traced, if necessary. It is not required that all of the metadata be maintained in this instance because of the possible size and management implications.

The metadata associated with the embedded contributor document content should include at a minimum:

- xapMM:DocumentID: set to the value of xapMM:DocumentID from the contributor document if one was present, otherwise set to a value based on the URL of the contributor document filename.
- xapMM:VersionID if there was a xapMM:VersionID present in the original.

- xapMM:RenditionClass should be set to a new value by the aggregating application. The value should be consistent with either the component's previous location or the component's new context. Examples:
  - "Figure 15" for the placement of Figure 15 of a document into the new one (Placed component was "Figure 15" of the source document).
  - "Image from page 15" for the placement of an image that was on page 15 into a new document (assuming the image didn't have its own document id)
- xapMM:History: The value from the contributor document, if any, can be ignored and the value set to a new single entry of the form: (action = "RenditionCreated," softwareAgent = "the aggregating app," when = "current date/time," parameters = "additional information the aggregating app chooses to include").
- xapMM:LastURL should be set to the value of xapMM:LastURL from the contributor document if one was present, or else set to the contributor document filename.
- xapMM:RenditionOf should be set to reference the contributor document. The reference should be to the document, version, and rendition class of the contributor. If those properties are not set in the contributor document, a URL for the contributor filename should be used as the document id in the reference.

In addition, in the metadata for the aggregate document, xapMM:ContainedResources should include a reference to each embedded resource.

If possible, the following additional metadata should be set in the metadata associated with the embedded content from the contributor document:

- xap:Author
- Applicable properties from xapG, xapGImg, xapDyn, xapDynA, xapDynV, xapT, and xapTPg schemas should be set to values consistent with the placed content. These values may be the same as those from the contributor document or may be different if the contributor document is modified or the corresponding metadata properties were not present or were incorrect in the contributor document.
- Applicable properties from the xapRights schema: set values for properties for which you have appropriate data

#### 7.3.3.4 Small Subset Embedding

When a small subset of the contributor document content is embedded in an undistinguished stream in the aggregate document, then no metadata is associated with the placed content.

Examples of this form would include embedding a copy of a few words or paragraphs of text or a few frames of video content from a contributor document into an aggregate document.

#### 7.3.3.5 Embedding of a Document Which Had No Metadata

When a contributor document has no metadata the application may include no metadata with the embedded placed content in the aggregate document. The application may set some metadata consistent with the previous sections if it determines that this would be in the user's best interest.

# A PDF and Dublin Core Schema

This appendix contains the schemas for the PDF and Dublin Core schemas. Aliases to XAP core metadata properties are specified where they apply.

## A.1 Adobe PDF Schema

***TABLE A.1 Adobe PDF Schema***

*The namespace prefix is pdf. The namespace is http://ns.adobe.com/pdf/1.3/.*

| Property | Value Type | Description | Alias Of | Category |
|---|---|---|---|---|
| pdf:Author | ProperName | Document author | xap:Author | External |
| pdf:BaseURL | URL | base URL for relative URLs | xap:BaseURL | Relational |
| pdf:CreationDate | Date | Document creation time | xap:CreateDate | Internal |
| pdf:Creator | AgentName | Tool that created the resource | xap:CreatorTool | Internal |
| pdf:Keywords | bag Text | Keywords | xap:Keywords | External |
| pdf:ModDate | Date | Last modify date | xap:MetadataDate | Internal |
| pdf:PDFVersion | Text | PDF file version (for example: 1.0, 1.3, etc.) | | Internal |
| pdf:Producer | AgentName | Name of tool that created PDF document | xap:CreatorTool | Internal |
| pdf:Subject | Text | Document subject text | xap:Description/*[@xml:lang= ‘x-default’] | External |
| pdf:Title | Text | Document title text | xap:Title/*[@xml:lang= ‘x-default’] | External |

## A.2 Dublin Core Schema

***TABLE A.2 Dublin Core Schema***

*The namespace prefix is dc. The namespace is http://purl.org/dc/elements/1.1/.*

| Property | Value Type | Description | Alias Of | Category |
|---|---|---|---|---|
| dc:contributor | bag ProperName | Other contributors to document | | External |
| dc:coverage | Text | The extent or scope of the resource | | External |
| dc:creator | bag ProperName | Authors who created document | xap:Author | External |
| dc:date | seq Date | Date(s) that something interesting happened to the resource | | External |
| dc:description | alt Text | A textual description of the resource, selected by language | xap:Description | External |
| dc:format | MIMEType | Data format | xap:Format | Internal |
| dc:identifier | Text | Unique identifier of the resource | | External |
| dc:language | alt Locale | Language(s) of the intellectual content | xap:Locale | Internal |
| dc:publisher | bag ProperName | Publishers | | External |
| dc:relation | bag Text | Relationships to other documents | | |
| dc:rights | alt Text | Informal rights statement, selected by language | xapRights:Copyright | External |
| dc:source | Text | Unique identifier of the work from which this resource was derived | | External |
| dc:subject | bag Text | The topic of the resource | xap:Keywords | External |
| dc:title | alt Text | Document title | xap:Title | External |
| dc:type | bag XChoice (open) | novel, poem, working paper, etc. | | External |